arXiv:2602.10320v1 [cs.FL] 10 Feb 2026

模网络架构的全局分布式协议可实现性

Elaine Li，纽约大学，美国
Thomas Wies，纽约大学，美国

全局协议从宏观视角描述分布式消息传递协议，并用作合成局部实现的规范。可实现性研究给定的全局协议是否存在分布式实现。我们首次全面研究了模网络架构的全局协议可实现性。我们提出一组网络参数化的一致性条件，并展示了在何种充分假设下，这些条件能够精确刻画可实现性。我们进一步将这些假设归约为一组最小的操作公理，用以描述各个消息缓冲区的插入和移除行为。通过这一归约，我们立即证明了五种常见的异步网络架构——点对点 FIFO、邮箱、发送方邮箱、单一邮箱和包——都是我们的网络参数化结果的实例。我们利用这一刻画，推导出模网络可实现性的最优复杂性结果、可实现全局协议类别之间的关系，以及用于判定模网络可实现性的符号算法。我们在首个网络参数化工具 Sprout(𝔸) 中实现了后者，并证明该工具在不牺牲性能和模块化的前提下实现了网络通用性。

附加关键词与短语：异步通信、网络语义、全局协议验证、通信状态机、多方会话类型

1 引言

分布式消息传递协议众所周知难以正确实现。异步网络中独立事件的交错使得检测孤儿消息、未指定接收和死锁等通信错误尤具挑战性。全局协议规范带有同步的假象：从全知观察者的视角出发，将发送与接收事件原子化地加以规定，从而通过构造排除大类通信错误。随后，全局协议被用作蓝图，以合成正确性由构造保证的分布式实现。这种方法启发了一种自顶向下的验证方法，其核心判定问题是可实现性，也称为可实现性。可实现性关注的是：给定的全局规范是否存在一个分布式实现；换言之，每个参与者的局部视图是否足以让所有参与者共同遵循协议设计者的全知意图，而不依赖隐蔽协调。

全局协议作为一种规范机制，并未预先限定特定的网络模型。相反，这一选择被推迟到为全局协议赋予语义时，而这又取决于目标分布式实现模型。迄今为止，文献中研究最多的网络模型是点对点（p2p）先进先出通信，其中协议参与者两两通过有序信道连接。针对点对点先进先出通信的全局协议的可实现性，已在高层消息时序图（HMSC）领域得到深入研究 [2, 3, 26, 28–30, 58, 61, 63, 64, 73]，点对点先进先出也是多方会话类型 [4, 5, 36–38, 42, 47, 53, 53, 76, 83] 和编舞式编程框架 [16, 33, 39, 40, 74]中占主导地位的网络模型。这类框架的实现已见于十多种编程语言。点对点先进先出通信在实践中无处不在，但其他网络模型也具有实践和理论上的研究价值。例如，Erlang 和 Go 实现了邮箱通信，而许多经典分布式算法（如领导者选举 [24] 和时钟同步 [50] ）的正确性证明依赖于袋语义。

作者联系信息：Elaine Li，纽约大学，纽约，美国，efl9013@nyu.edu；Thomas Wies，纽约大学，纽约，美国，wies@cs.nyu.edu。

自然会引出这样一个问题：全局协议的自顶向下方法能否扩展到面向不同的网络模型。通信模型的多样性及其语义定义方式，已作为一个独立研究主题得到探讨 [12, 13, 34], 但更重要的研究问题是，针对某一特定网络模型的现有结果能否推广到其他模型。在多网络场景下研究过的判定问题包括通信有限状态机的可同步性和可达性 [6, 19, 23, 58] 。某种网络模型中的通信错误不一定会在另一种模型中出现，因此设计真正网络参数化的算法极具挑战性。因此，绝大多数现有研究结果都针对固定的网络模型，而它们能否以及如何推广仍是未解问题。

我们通过首次对网络架构下全局规范的可实现性问题进行全面研究，对这一问题给出了肯定的回答。我们将网络架构表示为形式语言，并提出一组语义条件，称为广义一致性条件，其参数取决于所选择的网络架构。我们给出了充分条件，在这些条件下，我们的网络参数化条件相对于可实现性是可靠且完备的。这些充分条件表现为形式语言上的一组抽象的代数性质，用于定义网络架构的行为，也是实现统一的、网络参数化的可靠性与完备性证明的关键。这组首要贡献构成了针对全局协议可实现性的精确网络参数化解决方案。

我们进一步推进了网络参数化，将基于形式语言理论的充分性条件归约为一组定义在缓冲区数据结构上的简单操作公理。这些缓冲区公理大大简化了确定某一网络架构是否属于我们结果实例的任务，因为该任务归结为证明关于插入和移除操作的少数规范。作为这一归约的直接结果，我们得出结论：五种通常考虑的网络模型，即点对点 FIFO、邮箱（n-1）、发送方邮箱（1-n）、单邮箱（1-1）和包，均属于我们的参数化结果实例。一个或许更令人惊讶的结果是，我们的公理完全不约束通信拓扑，甚至能够适用于包含多个缓冲区数据结构的异构网络架构。

作为最后一项贡献，我们推导出用于检查协议相对于网络架构的可实现性的具体算法，并针对多种协议片段得出可判定性和复杂性结果。对于有限全局协议，我们证明，在上述所有网络架构下，可实现性问题都是 co-NP 完全，即使全局协议属于有向选择多方会话类型片段 [42]这一特殊情形也是如此。对于带有一阶逻辑转移约束的符号化全局协议，我们证明，我们的网络参数化可实现性刻画可以编码为一阶定点逻辑 μCLP [78]。

因此，我们的刻画给出了一个符号化算法，用于检查具有无限多个状态和数据的协议的可实现性。我们将推导出的符号化算法实现为工具 Sprout（𝔸），并扩展了一款现有工具；该工具可在假设网络为点对点 FIFO 网络 [56]的情况下判定全局协议的可实现性。我们的评估表明，Sprout（𝔸）可以模块化地扩展到不同架构，同时不牺牲性能。

贡献。总而言之，我们的贡献如下：

- 我们提出了全局协议规范相对于网络架构的可实现性问题。
- 我们给出以网络参数化方式刻画可实现性的条件，并提供使该刻画可靠且完备的充分条件。
- 我们将这些充分条件归约为一个简单的缓冲区数据结构公理化模型，该模型可以针对任何给定的网络架构轻松进行检验。

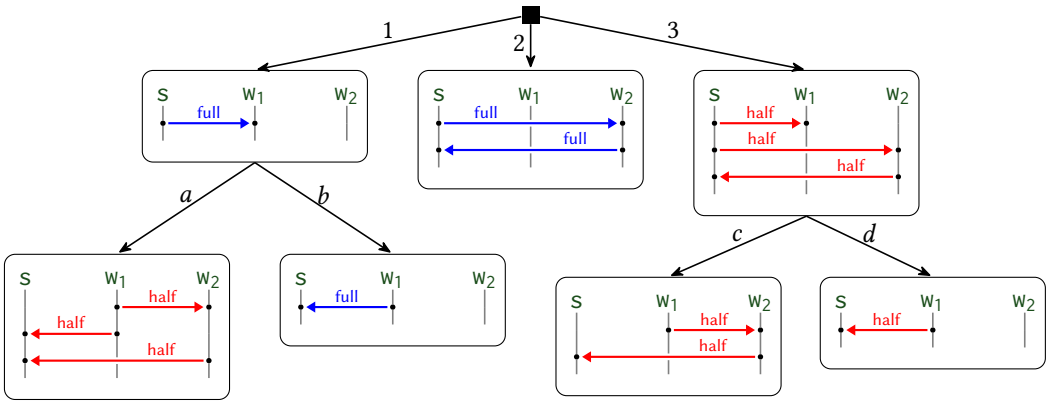


图 1. 带任务委托的任务调度器。

- 我们针对五种常见的网络架构推导出有限状态协议可实现性的可判定性和复杂度结果，并给出了检验符号全局协议可实现性的算法。
- 我们介绍 Sprout(𝔸)，这是首个面向符号全局协议的网络参数化可实现性检查器。

2 动机示例

我们使用任务调度协议来说明不同网络架构之间的细微差异。该协议的全局规范如图 1 所示，采用高层消息时序图（HMSC）表示法。该协议涉及三个参与者：调度器 s 以及两个工作者 w_1 和 w_2。最初，s 选择将整个任务交给 w_1，或仅交给 w_2，，也可以决定在两个工作者之间拆分工作负载。前两种情况分别显示在分支 1 和分支 2 中，其中 s 向相应的工作者发送一条完整消息。第三种情况是分支 3，此时 s 向两个工作者都发送一条半量消息。工作者 w_2 始终立即完成分配给它的任务，并通过回显从 s 收到的消息将结果发送回 s。然而，每当 w_1 被分配任务时，它可以选择像 w_2 一样行动（分支 b 和 d），也可以将部分或全部工作委托给 w_2（分支 a 和 c）。该协议在循环中运行，但为便于阅读，我们在图 1 中省略了指向初始状态的回边。

模网络架构的可实现性。 可实现性探讨的是：对于三个参与者，是否存在局部实现，使其在异步网络架构上并发执行时，能够按照全局协议规范行事。具体而言，实现绝不能发生死锁，并且所有参与者都应根据各自局部选择的分支保持一致，严格按照规定的顺序执行发送和接收动作。后一项性质称为协议保真性。网络架构作为问题陈述中的一个参数保留。

判定可实现性。 局部实现只能通过作出分支决策和接收消息来获取有关全局协议状态的信息。当参与者的局部信息（包括迄今为止所作的决策和获得的观察）不足以确定其下一步动作时，就会出现不可实现性。

让我们先分析任务调度协议在标准点对点先进先出网络架构（也称为点对点盒语义）下的可实现性问题

，本文中称为）。在全文中，我们用$p \triangleright q!m$表示一个发送事件，其中参与者p向m发送消息给q。同样，我们用$q \triangleleft p?m$表示一个事件，其中q接收消息m，该消息先前由p发送。

在点对点盒式网络下，该协议不可实现：w_2无法区分遵循分支 1 和 a的运行与遵循分支 3 和 c的运行。在这两种情况下，w_1都可能处于相同的局部状态q，此时其关联的信道缓冲区中只有来自w_1的半条消息可用（也就是说，在 $3c$运行中，从 s 到w_1的半条消息可能会被延迟）。如果协议遵循$1a$运行，w_2的下一步操作应当是向 s 发送回复。然而，在 $3c$运行中，它应当先等待来自 s 的半条消息到达。如果w_2在状态q中等待，这将在 $1a$运行中导致死锁；而如果它向 s 发送消息，则会在 $3c$运行中违反协议保真性。

或许令人意外的是，用另一种异步网络架构替代点对点盒式网络并不能解决这一问题。不可实现性的原因完全取决于通信的异步性，以及以下两个发送事件$s \triangleright w_2!\mathrm{half}$和$w_1 \triangleright w_2!\mathrm{half}$在 $3a$运行中并不具有因果依赖关系。因此，它们可以并发发生，从而使两条消息以任意顺序到达w_2。因此，对于任何异步网络架构，该协议都是不可实现的。

然而，一般而言，可实现性取决于具体的网络架构。例如，考虑对全局规范进行一种可能的修复：将分支 half 上从w_1发送给w_2的消息值替换为 $1a$ delegate。现在，w_2就能区分这两个分支：它可以等待，直到缓冲区中出现来自 half s的消息，表明协议遵循分支 3；或者等待缓冲区中出现来自delegate w_2, 的消息，表明其他参与者选择了分支$1a$。由于这两种情况互斥，w_2一旦观察到其中一种情况，就能立即作出决定。这一改变使该协议在点对点盒语义下可实现。

另一方面，这种修复对邮箱网络架构并无帮助，因为在该架构中，发送给同一收件人的所有消息都会被收集到一个先进先出缓冲区中。邮箱的问题在于，在 $3c$分支中，网络仍可能通过延迟来自 s 的消息，异步地重新排序发送给w_2的两条消息。由于消息按照到达顺序存入先进先出缓冲区，这会迫使w_2先接收来自w_1的消息，然后才能从缓冲区中取出来自 s 的消息。由此产生的事件序列会违反协议保真性。请注意，这次协议保真性的违反与协议参与者对协议正在遵循哪个分支的信息不完整无关，而完全是由于网络架构能够在单个协议运行的执行过程中重新排序彼此独立的事件。针对这种架构，一种可能的协议修复方式是在$s \triangleright w_2!\mathrm{half}$和$w_1 \triangleright w_2!\mathrm{half}$之间引入因果依赖，例如在这两次交换之间插入额外的消息交换$s \triangleright w_1!\mathrm{done}$，迫使$w_1$等待，直到 s 已向$w_2$发送消息。

最后，如果我们将网络架构从邮箱切换为邮件袋，即用无序多重集替代每个收件人的单个先进先出队列，那么即使没有 proposedsecondrepairformailbox，协议也再次变得可实现。

3 预备知识

词。设 Σ 为一个字母表。Σ^* 表示由 Σ构成的有限字集合，Σ^ω 表示无穷字集合，而 Σ^∞ 表示二者的并集 $\Sigma^* \cup \Sigma^\omega$。若存在 $w \in \Sigma^\infty$ 满足 $u \cdot w = v$，则字 $u \in \Sigma^*$ 是字 $v \in \Sigma^\infty$的前缀，记作 $u \le v$；所有 u的前缀记作 $\mathrm{pref}(u)$。有时我们省略连接符号 $\cdot$，改写为 $uw = v$。给定一个字 $w = w_0 \dots w_n$，我们用 $w[i]$表示第 i个符号 $w_i \in \Sigma$，用 $w[0..i]$表示包含 w_0和 w_i在内的子词，即 $w_0 \dots w_i$。

消息字母表。设 P 是一个（可能无限的）参与者集合，V 是一个（可能无限的）数据域。我们定义同步事件集合 $\Gamma_{sync} := \{p \to q : m \mid p, q \in P \text{ and } m \in V\}$，其中 $p \to q : m$ 表示从发送方 p 到接收方 q 的一条消息交换。对于参与者 $p \in P$，我们定义字母表 $\Gamma_p = \{p \to q : m \mid q \in P \quad m \in V\} \cup \{q \to p : m \mid q \in P \quad m \in V\}$，以及一个同态 $\Downarrow_{\Gamma_p}$，其中 $x\Downarrow_{\Gamma_p} = x$ if $x \in \Gamma_p$，否则为 ε。对于参与者 $p \in P$，我们定义发送事件字母表 $\Sigma_{p,!} = \{p \triangleright q!m \mid q \in P \quad m \in V\}$，以及接收事件字母表 $\Sigma_{p,?} = \{p \triangleleft q?m \mid q \in P \quad m \in V\}$。事件 $p \triangleright q!m$ 表示参与者 p 向 q 发送消息 m，而 $p \triangleleft q?m$ 表示参与者 p 接收来自 q 的消息 m。我们记 $\Sigma_p = \Sigma_{p,!} \cup \Sigma_{p,?}$, $\Sigma_! = \bigcup_{p \in P} \Sigma_{p,!}$，以及 $\Sigma_? = \bigcup_{p \in P} \Sigma_{p,?}$。最后，异步事件集合为 $\Sigma_{async} = \Sigma_! \cup \Sigma_?$。

投影。我们称 p 在 $x \in \Sigma_{async}$ if $x \in \Sigma_p$ 中处于活动状态。对于每个参与者 $p \in P$，我们定义一个同态 $\Downarrow_{\Sigma_p}$，其中 $x\Downarrow_{\Sigma_p} = x$ if $x \in \Sigma_p$，否则为 ε。我们基于字母表符号的模式匹配定义一类投影，记作 $\Downarrow$-。投影结果由模式中未指定的部分决定。例如，$\Downarrow_{p\triangleright\text{-}!\text{-}}$ 将符号 $p \triangleright q!m$ 投影为 $(q\ m)$，而非发送符号以及发送方不是 p 的发送符号则投影为 ε。函数 $\Downarrow_{q\triangleleft p?\text{-}}$ 将 p 从 q 接收的任意消息值的接收事件投影为该消息值，并将所有其他事件投影为 ε。

我们采用基于同步字母表 Γ_{sync} 的标记迁移系统，作为描述全局协议的起点。

标记迁移系统。A 标记迁移系统(LTS) 是一个元组 $S = (S,\ \Gamma,\ T,\ s_0,\ F)$，其中 S 是状态集，Γ 是标签集，T 是从 $S \times \Gamma \times S$ 出发的迁移集，$F \subseteq S$ 是终态集，$s_0 \in S$ 是初始状态。我们用 $p \xrightarrow{\alpha} q$ 表示迁移 $(p,\ \alpha,\ q) \in T$。标记迁移系统的运行和轨迹按通常方式定义。若一次运行要么是有限的且以终态结束，要么是无限的，则称其为极大。标记迁移系统 S 的语言（记为 L(S)）定义为极大轨迹的集合。若状态 $s \in S$ 不是终态且没有出向转移，则称其为死锁。若没有可达状态是死锁，则该标记迁移系统是无死锁的。

在 [54], 上的标记迁移系统 Γ_{sync} 受到三个额外条件的约束，从而形成一个称为全局通信标记迁移系统的片段，以下称为 GCLTS。三个 GCLTS 假设是汇点终态性、发送方驱动选择和无死锁性。汇点终态性是一个纯粹的语法条件，要求终态没有出向转移。发送方驱动选择规定，从任意状态出发的所有出向转移都共享唯一的发送方，而且具有确定性。也就是说，在协议的任何分支点，都有唯一的发送方决定采取哪个分支。文献 [58, 75] 指出，放弃发送方驱动选择会导致可实现性不可判定，尽管近期人们对同步协议片段中的混合选择产生了兴趣 [72]。无死锁性要求所有可达状态要么是终态，要么具有出向转移。

我们的任务调度示例（图1）满足所有 GCLTS 假设。特别是，要看出它满足发送方驱动选择，只需注意 s 在分支 {1,2,3} 中进行选择，而 w_1 分别在 $\{a, b\}$ 和 $\{c, d\}$ 中进行选择。

4 模网络架构的可实现性

本节将介绍分布式实现、全局协议语义的网络参数化定义，最后介绍可实现性问题。

我们的实现模型基于通信状态机（CSM） [8]。通信状态机由一组有限状态机组成，每个参与者对应一个有限状态机，并通过成对先进先出信道进行通信。我们从两个关键维度对通信状态机进行了泛化：通信拓扑，以及

消息缓冲区的数据结构。我们还取消了通信状态机施加的限制，即参与者的数量以及局部状态机的状态空间必须是有限的，遵循 [54, 57]。

定义 4.1（网络架构）。参与者集合 P 和消息值集合 V 上的网络架构是一个元组 $\mathbb{A} = (C, B, ch, ins, rem\ b_0)$，其中 C 是信道集合，B 是信道内容集合，而 $ch : P \times P \to C$ 是一个将每个发送方和接收方与一个信道关联起来的映射。直观地说，ch(p, q) 表示用于存放从 p 发送到 q 的消息的消息缓冲区。我们将 ch 称为 $\mathbb{A}$ 的通信拓扑。信道状态集合为 $\mathbb{X} = C \to B$。

信道内容中的消息 $x \in M$ 带有其发送方和接收方的标记，即 $M = P \times P \times V$。信道内容配备了部分插入和移除操作，$ins,rem : M \to B \to B$，其中，$ins(x)(b)$ 未定义表示 b 在插入 x 时阻塞，而仅当 x 可在 b 中被移除时，$rem(x)(b)$ 才有定义。最后，$b_0 \in B$ 是空信道内容。

例 *4.2*。我们定义了八种具体的网络架构，稍后将再次讨论。我们考虑四种通信拓扑：n 对 n，即所有发送方和接收方共用同一信道；一对 n，即接收方共用同一信道，从单个发送方接收消息；n 对一，即发送方共用同一信道，向单个接收方发送消息；以及一对一，即每对发送方和接收方都有一个唯一的信道。我们考虑两种消息缓冲区数据结构：有序的 FIFO队列和无序多重集。上述网络架构常见的名称包括全局总线（n 对 n FIFO）、消息池（一对一多重集）和邮箱（n 对一 FIFO）。消息池常用于 Paxos 和 Raft 等领导者选举协议；一对 n FIFO 则常见于并行编程中的工作窃取模式。最后，一对一或点对点 FIFO 是通信状态机的标准网络架构，在消息传递并发的理论与实践中也被广泛采用。

四种通信拓扑定义如下（括号中为我们采用的命名约定，其中“B”指下列某种缓冲区类型的名称）：

- n 对 n（点对点 B）：$C = \mathcal{P} \times \mathcal{P}$，$ch(p, q) = (p, q)$，
- 1 对 n（邮件 B）：$C = \mathcal{P}$，$ch(p, q) = p$，
- n 对 1（发送方 B）：$C = \mathcal{P}$，$ch(p, q) = q$，
- 1 对 1（单 B）：$C = \{0\}$，$ch(p, q) = 0$，

而两种缓冲区类型为先进先出队列（B=包），$B = M^*$，以及多重集（B=包），$B = M \to \mathbb{N}$。对于先进先出队列，插入对应于在队列末尾追加，移除对应于从队列头部移除；对于多重集，插入表示多重集加法，而移除表示多重集删除。空缓冲区内容 b_0 在先进先出队列的情况下为 ε，在多重集缓冲区的情况下为 $\emptyset$。

我们注意到，具有同质袋通道的四种网络架构在操作上等价，并可归并为单袋情形：由于消息都标记有其发送方和接收方，因此每当发送或接收消息时，都可以“按需”将消息池拆分为 P^2 个多重集，或按发送方或接收方划分为 P 个多重集，从而模拟其他网络架构。因此，剩下的是四种先进先出网络架构，本文后续将其称为 p2p、mb、sb 和 monob，以及袋式通道归并后的情形（bag）。

接下来，我们介绍以网络架构 $\mathbb{A}$ 的选择为参数的通信标记迁移系统。对于以下定义，我们固定一个网络架构 $\mathbb{A} = (C, B, ch, ins, rem\ b_0)$。我们通过如下定义，将插入和移除操作提升到 $\mathbb{A}$ 的信道状态 $\xi \in \mathbb{X}$：$insert(\xi, p, q\ m) = \xi'$，其中 $\xi'(ch(p, q)) = ins(p, q\ m)(\xi(ch(p, q)))$，而其他所有信道内容均保持不变；$remove(\xi, p, q\ m)$ 的定义类似。

定义 *4.3*（网络参数化通信标记迁移系统）。$\mathcal{T}_{\mathbb{A}} = \{\{T_p\}\}_{p \in \mathcal{P}}$ 是通信标记迁移系统（CLTS），定义在$\mathcal{P}$、$\mathcal{V}$ 和 $\mathbb{A}$，当且仅当 T_p 是定义在 Σ_p 上的确定性标记迁移系统，对每个$p \in \mathcal{P}$，记作

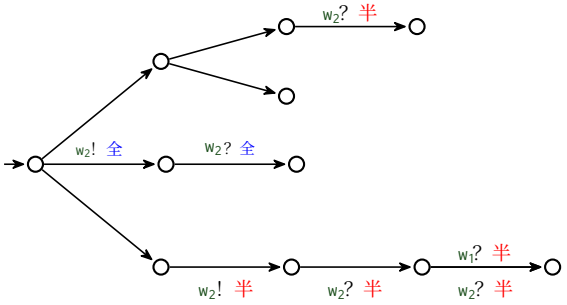


图 2. 局部实现对应s

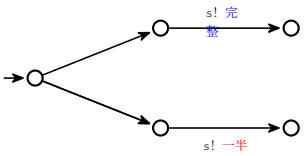


图 3. 局部实现对应w_2

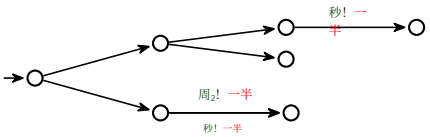


图 4. 局部实现周$_1$

$(Q_p, \Sigma_p, \delta_p, q_{0,p}, F_p)$。令 $\prod_{p \in P} Q_p$ 表示全局状态的集合。$T_{\mathbb{A}}$ 的一个配置是一个二元组 $(\vec{s}, \xi)$，其中 $\vec{s}$ 是全局状态，$\xi \in \mathbb{X}$ 是信道状态。我们使用 $\vec{s}_p$ 表示 $\vec{s}$ 中 p 的状态。CLTS 转移关系记作 $\rightarrow$，定义如下。

- $(\vec{s}, \xi) \xrightarrow{p \triangleright q!m} (\vec{s}', \xi')$ if $(\vec{s}_p, p \triangleright q!m, \vec{s}'_p) \in \delta_p$, $\vec{s}_r = \vec{s}'_r$ for every participant $r \neq p$, $\xi' = insert(\xi, p, q, m)$.
- $(\vec{s}, \xi) \xrightarrow{q \triangleleft p?m} (\vec{s}', \xi')$ if $(\vec{s}_q, q \triangleleft p?m, \vec{s}'_q) \in \delta_q$, $\vec{s}_r = \vec{s}'_r$ for every participant $r \neq q$, $\xi' = remove(\xi, p, q, m)$.

在初始配置 $(\vec{s}_0, \xi_0)$ 中，$\vec{s}_0$ 内每个参与者的状态都是 A_p的初始状态$q_{0,p}$，而 ξ_0将每个信道映射到空缓冲区 b_0。当且仅当对每个 p，$\vec{s}_p$ 都是终结配置且 $\xi = \xi_0$时，配置 $(\vec{s}, \xi)$ 才是终结配置。运行和轨迹按预期方式定义。如果运行是有限的且以终结配置结束，或者是无限的，则称其为极大运行。CLTS $T_{\mathbb{A}}$ 的语言 $L(T_{\mathbb{A}})$定义为极大轨迹的集合，而 $\mathrm{pref}(L(T_{\mathbb{A}}))$定义为极大轨迹前缀的集合。如果配置 $(\vec{s}, \xi)$ 不是终结配置且没有出向转移，则称其为死锁。如果一个 CLTS 没有可达配置处于死锁，则称其为无死锁。

参与者的局部实现 s、w_1 和 w_2 §2 中修复后的任务调度协议的实现如图 2、图 4 和图 3 所示。为清晰起见，转移标签中省略了活跃参与者。

信道符合性。在定义全局协议的语义之前，先将给定网络架构 $\mathbb{A}$ 的行为与任何特定 CLTS $T_{\mathbb{A}}$的行为区分开来是很有用的。为此，我们引入信道符合性这一语义概念，用以描述与 $\mathbb{A}$一致的所有词 w。信道一致词是通用 CLTS 的轨迹；通用 CLTS 放宽了参与者局部状态所施加的所有轨迹约束。形式化地，令 $U_{\mathbb{A}}$ 为 $\mathbb{A}$的通用 CLTS，其中 $U_{\mathbb{A}} = \{\{U_p\}\}_{p \in P}$，且对每个 $p \in P$ 都有 $L(U_p) = \Sigma_p^*$。若 $w \in \Sigma_{async}^*$是$\mathbb{A}$-信道合规的，当且仅当 w 是 $U_{\mathbb{A}}$的一条轨迹。若 $\mathbb{A}$ 已明确，我们就简称 w 信道合规。此外，若实际上 $w \in L(U_{\mathbb{A}})$成立，则称 w 为信道匹配的。直观地说，如果能在 $\mathbb{A}$的任意给定 CLTS 中执行一个信道匹配词，那么在到达的配置中，所有缓冲区都将为空。我们用 $L(\mathbb{A}) \subseteq \Sigma_{async}^*$ 表示所有信道一致词。

局部投影下的相等。我们称 w_1 和 w_2 在局部投影下相等，记作$w_1 \equiv_{\mathcal{P}} w_2$，如果对所有 p，$w_1 \Downarrow_{\Sigma_p} = w_2 \Downarrow_{\Sigma_p}$。我们用 $[w]_{\equiv_{\mathcal{P}}}$ 表示局部投影下以 w为代表元的等价类。我们将其推广到集合 $W \subseteq \Sigma_{async}^{\infty}$，定义为 $[W]_{\equiv_{\mathcal{P}}} = \bigcup_{w \in W} [w]_{\equiv_{\mathcal{P}}}$。

全局协议语义。我们定义全局协议 S 的异步语义，记作 $\mathrm{L}_{\mathbb{A}}(\mathrm{S}) \subseteq \Sigma^{\infty}_{async}$，并以所选的网络架构 $\mathbb{A}$ 为模。回顾一下，S 是一个定义在字母表 Γ_{sync} 上的标记迁移系统。语义 $\mathrm{L}_{\mathbb{A}}$(S) 的出发点是同步语言 L(S)。从 L(S) 出发，我们可以通过拆分获得一组 1-同步异步字；拆分操作会将每个原子发送与接收事件拆分为对应的两个事件，记作 split(L(S))。我们希望纳入所有在局部投影下、相对于给定网络架构 $\mathbb{A}$与这些 1-同步词相等的异步字。

我们分别处理有限字和无穷字，将全局协议语义定义为其有限语义与无穷语义的并集：

$$\mathcal{L}_{\mathbb{A}}(\mathcal{S}) = \mathcal{L}^{\mathrm{fin}}_{\mathbb{A}}(\mathcal{S}) \cup \mathcal{L}^{\mathrm{inf}}_{\mathbb{A}}(\mathcal{S})$$

按照上述方法，并将 split(L (S)) 限定为有限字，可得到有限语义：

$$\mathcal{L}^{\mathrm{fin}}_{\mathbb{A}}(\mathcal{S}) = [\Sigma^{*}_{async} \cap \mathtt{split}(\mathcal{L}(\mathcal{S}))]_{\equiv_{\mathcal{P}}} \cap \mathcal{L}(\mathbb{A}) \ .$$

无穷语义是指这样的字：它们的前缀在局部投影下相等且符合网络语义的意义下，可以扩展为 L(S) 中的某个字：

$$\mathcal{L}^{\mathrm{inf}}_{\mathbb{A}}(\mathcal{S}) = \{w \in \Sigma^{\infty}_{async} \mid \forall u \leq w.\, u \in \mathrm{pref}([\mathtt{split}(\mathcal{L}(\mathcal{S}))]_{\equiv_{\mathcal{P}}} \cap \mathcal{L}(\mathbb{A}))\} \ .$$

为避免歧义，我们将 $\mathrm{L(S)} \subseteq \Gamma^{\omega}_{sync}$称为 S 的*LTS* 语义，并将 $\mathrm{L}_{\mathbb{A}}\mathrm{(S)} \subseteq \Sigma^{\omega}_{async}$称为 S 的协议语义。

例 4.4. 为了说明我们的协议语义，考虑以下协议，其语义中同时包含有限词和无穷词：$\mathrm{p} \to \mathrm{q} : m^{\infty} + (\mathrm{p} \to \mathrm{q} : m)^{*} \cdot \mathrm{r} \to \mathrm{q} : m$。该协议的同步运行要么形式为 $\mathrm{p} \to \mathrm{q} : m^{\infty}$，要么形式为 $(\mathrm{p} \to \mathrm{q} : m)^{n} \cdot \mathrm{r} \to \mathrm{q} : m$。拆分运行随后形式为 $(\mathrm{p} \triangleright \mathrm{q}!m \cdot \mathrm{q} \triangleleft \mathrm{p}?m)^{\infty}$ 或 $(\mathrm{p} \triangleright \mathrm{q}!m \cdot \mathrm{q} \triangleleft \mathrm{p}?m)^{n} \cdot \mathrm{r} \triangleright \mathrm{q}!m \cdot \mathrm{q} \triangleleft \mathrm{r}?m$。由于我们的无限词语义不施加任何公平性假设，未公平调度的词 $\mathrm{p} \triangleright \mathrm{q}!m^{\infty}$属于该协议的无穷语义。词 $\mathrm{r} \triangleright \mathrm{q}!m \cdot \mathrm{p} \triangleright \mathrm{q}!m \cdot \mathrm{q} \triangleleft \mathrm{p}?m \cdot \mathrm{q} \triangleleft \mathrm{r}?m$属于点对点条件下该协议的有限语义；在此情况下，网络将来自 p 和 r 的发送事件重新排序，而 q 按照协议规定的顺序接收消息，先接收来自 p 的消息，再接收来自 r 的消息。

现在，我们可以定义网络参数化可实现性问题：

定义 *4.5*（网络参数化协议可实现性）。如果存在一个 CLTS $\mathrm{T}_{\mathbb{A}} = \{\{T_{\mathrm{p}}\}\}_{\mathrm{p} \in \mathcal{P}}$，使得以下两个性质成立，则称协议 S 在网络架构 $\mathbb{A}$ 下是可实现的：(i) 协议保真性：$\mathrm{L}(\{\{T_{\mathrm{p}}\}\}_{\mathrm{p} \in \mathcal{P}}) = \mathrm{L}_{\mathbb{A}}\mathrm{(S)}$；以及 (ii) 无死锁性：$\{\{T_{\mathrm{p}}\}\}_{\mathrm{p} \in \mathcal{P}}$无死锁。我们称 $\{\{T_{\mathrm{p}}\}\}_{\mathrm{p} \in \mathcal{P}}$ 在 $\mathbb{A}$ 下实现了 S。

5 广义可实现性的刻画

本节将介绍我们提出的可靠且完备的网络参数化可实现性刻画。我们的出发点是近期提出的一种针对点对点可实现性 [54]的精确刻画。我们阐明了 [54] 中的刻画所采用的、关于协议语义和实现模型的关键抽象假设，以及这些假设如何使其可靠性和完备性证明得以成立。类似于计算最弱前置条件的过程，我们在构建网络参数化刻画的同时，提炼并在某些情况下弱化这些抽象假设。最终，我们得到一组网络参数化条件，称为广义一致性条件；同时还得到一组抽象假设，在这些假设成立时，我们的刻画对于可实现性是可靠且完备的。

[54] 的可实现性刻画由三个一致性条件（一致性条件）构成，全局协议必须满足这些条件。一致性条件是二重超性质，它们审视参与者能够执行不同动作的全局协议状态对，而这些状态之间的区别对参与者而言可能无法在局部观察到，

但这些状态之间的区别对参与者而言可能无法在局部观察到。一致性条件 描述了在这种不知情状态下可以安全执行的局部动作类型。发送一致性 指出，如果参与者可以从某个状态执行发送动作，那么它也必须能够从任何不可区分的状态执行相同的发送动作。接收一致性 指出，如果参与者可以从某个状态执行接收动作，那么就不可能从任何其他不可区分的状态执行同一接收动作。无混合选择 指出，参与者不能在执行发送动作和接收动作之间含糊其辞。

形式化地说，全局协议状态 s 对于 参与者 在 $u \in \Gamma_{\mathsf{p}}^{*}$ 上是可达的，当且仅当存在 $w \in \Gamma_{sync}^{*}$ 使得 s 通过轨迹 w 和 $w{\Downarrow_{\Gamma_{\mathsf{p}}}} = u$ 可达。如果两个全局协议状态 s_1 和 s_2 对于参与者 在相同的 $u \in \Gamma_{\mathsf{p}}^{*}$ 上都可达，我们称它们为同时可达。同时可达性刻画了参与者在局部上无法区分的状态对。

Li 等人 [54]通过调用一致性条件，证明了其可靠性；其中规范实现充当可实现性的见证。

定义 *5.1*（规范实现 *[54]*）。若对每个 $p \in P$，T_{p} 均满足：*(i)* $\forall w \in \Sigma_{\mathsf{p}}^{*}.\ w \in L(T_{\mathsf{p}}) \Leftrightarrow w \in L(S){\Downarrow_{\Sigma_{\mathsf{p}}}}$，且 *(ii)* $pref(L(T_{\mathsf{p}})) = pref(L(S){\Downarrow_{\Sigma_{\mathsf{p}}}})$，则 $CLTS\ \{\{T_{\mathsf{p}}\}\}_{\mathsf{p}\in\mathcal{P}}$ 是协议 $S = (S, \Gamma_{sync}, T, s_0, F)$ 的规范实现。

正如 [57], 所观察到的，规范性可以直接根据全局协议的 LTS 语义来定义，因为局部投影不受异步重排的影响。正如 [54]所讨论并在 [57], 中形式化的那样，规范实现总能通过广义子集构造得到，而综合与实现所需的网络架构考量彼此独立。我们将在 §10中进一步讨论综合。

可靠性的关键技术论证在于证明规范实现的语言是全局协议语义的子集，且不存在死锁。这需要证明，每条规范实现轨迹都能与全局协议中的某个运行相关联，并且在该运行中，每个参与者都已部分完成规定的动作。这一点又通过对规范实现轨迹进行归纳法证明，并借助一致性条件 论证：无论扩展是通过发送事件还是接收事件进行，都能保留一个与所得轨迹相关联的全局运行。

在 [54] 中通过逆否推理确立一致性条件 的完备性：从每个一致性条件的否定出发，构造一条轨迹，该轨迹不符合任何协议运行，却必定会被该协议的任一候选实现接纳。这足以说明该协议违反可实现性，因为该轨迹要么导致死锁，要么导致一个不属于全局协议语义的极大词。

重新审视发送一致性与无混合选择。 这两个一致性条件在其原始形式下，对于网络参数化可实现性仍然可靠且完备。然而，它们的可靠性证明与完备性证明仍需推广到网络参数化情形。

为了推广发送一致性和无混合选择的完备性证明，我们注意到，[54] 中的完备性证明所构造的见证可以调整为 1-同步轨迹的一个前缀。具体而言，这两个条件的见证都具有 $\mathrm{split}(\alpha)\cdot x$ 的形式，其中 α 是同步词，x 是发送事件。随后，证明表明 $\mathrm{split}(\alpha)\cdot x$ 是信道合规的；再结合它必定可由任何候选实现执行这一事实，便可得到一个可实现性的反例。因此，这一推广首先要求假设所考虑网络架构中的发送转换始终处于启用状态。我们将这一假设形式化如下：

$$\forall w \in \Sigma_{async}^{*},\ x \in \Sigma_{!}.\ w \text{是信道合规的} \implies wx \text{是信道合规的。} \tag{F1}$$

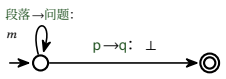


图 5. 一个无法在袋式网络上实现、但可在点对点盒式网络上实现的协议 S_a

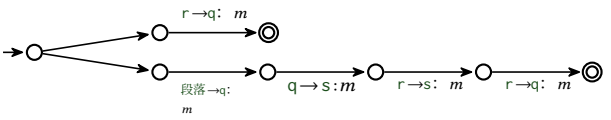


图 6. 一个协议 S_b，它无法在点对点盒式网络上实现，却可以在发送方盒式网络上实现。

为了完成网络参数化证明，我们证明 $\mathtt{split}(\alpha) \cdot x$ 是信道合规的。这一事实由 F1 得出，前提是 $\mathtt{split}(\alpha)$ 是信道合规的。这引出了我们的第二个假设：

$$\forall \alpha \in \Gamma^*_{sync}.\ \mathtt{split}(\alpha) \text{ is channel-compliant.} \tag{F3}$$

直观而言，合理的网络应允许发送及其对应的接收连续执行。发送一致性的可靠性旨在证明，规范实现进行的任何发送扩展仍然是协议前缀。事实证明，这一点既取决于无混合选择，也取决于接收一致性。接下来，我们将讨论如何推广接收一致性。

重新审视接收一致性。与发送事件不同，接收事件不仅取决于迁移，也取决于待接收消息的可用性。而消息可用性又高度依赖于网络架构，因此需要考虑可能的异步重排；任务委派协议中的参与者 w_2在 §2 中的处境便印证了这一点。

考虑协议 S_a 和 S_b，如图 5 和图 6 所示，两者均从接收方 q 的视角出发。在单袋网络架构下的 S_a 中，除了 ⊥ 消息外，q 的袋中还可以包含来自 p 的任意数量的m 消息。参与者 q 永远无法知道何时接收 ⊥ 消息才是安全的，因此该协议不可实现。如果将单袋网络替换为点对点盒式网络，那么最后一条 ⊥ 消息会排在所有 m 消息发送完毕之后，因此参与者 q 总能接收其来自 p 的 FIFO 队列队首消息。在点对点盒式网络下的 S_b 中，当来自 q 的消息 m 可供接收时，来自 p 的同一条消息 m 也可能同时可供接收，从而导致 q 违反协议。如果将点对点盒式网络替换为发送方盒式网络，这种歧义便会再次消失：r 发给 s 的消息实际上会阻止发给 q 的消息变为可接收，因此无论 p 选择哪个分支，q 都只有一条可接收的消息。

为了推广接收一致性，我们首先对全局协议的所有实现作出以下关键观察。如果在实现执行过程中的任何时刻，某个参与者都可以选择接收两条不同的消息，那么其中一个选择必然构成协议违反。由于协议运行对所有参与者的事件进行了全序排列，因此不存在一个同时符合这两种接收选择的协议运行。假设对于每种接收选择，都存在一个协议运行符合由此产生的执行轨迹。鉴于全局协议的发送方驱动性，对于全局协议的任意两个运行，它们首次出现差异的事件必定具有相同的发送方。接收方能够在两种接收之间作出选择，这意味着它无法察觉相应的合规运行之间的差异，因此它的选择会迫使发送方在两个分支中作出承诺。然而，参与者之间不可能进行这种协调，因为发送事件要么在因果关系上先于接收事件，要么与接收事件相互独立。事实上，[54] 中的接收一致性可靠性证明表明，规范实现的轨迹的所有接收扩展都是唯一的；而接收一致性的完备性证明则通过构造反例表明，只要存在一个具有两种不同接收扩展的实现前缀，该协议就不可实现。

由这一关键观察可知，无论网络架构如何，只需且必须保证：对于任意给定的参与者，在任何

配置中，都不会同时存在两个不同的接收。随后，我们以网络参数化的方式描述这一更简单的性质。关于 m 某个词执行后，在某个配置中可被q 从 p 接收 w 这一概念，只需通过 $w\cdot \mathrm{q} \triangleleft \mathrm{p}?m$的信道符合性来描述。我们还希望对 w施加另外两项约束：首先， m 由 p 发送，不应取决于 q执行的任何事件，因此我们要求 $w{\Downarrow_{\Sigma_\mathrm{q}}} = \varepsilon$；其次，还必须向q 发送过另一条不同的消息，且该消息位于 w中：发送方或消息值必须不同。

由此引出广义接收一致性条件，定义如下：

定义 5.2（广义接收一致性）。若对于任意两个由 q 同时可达的状态 s_1, s_1'，其迁移为 $s_1 \mathrm{p}\xrightarrow{\rightarrow\ m} s_2\mathrm{q}:$ 和 $s_1' \mathrm{r}\ \rightarrow\ m' \ s_2'\mathrm{q}:$，当 $\mathrm{r} \neq \mathrm{p}$ 或 $m' \neq m$ 时，对于任意 $w \in \mathrm{pref}\,(\mathrm{L}(\mathrm{S}^{s_1}))$，只要 $w{\Downarrow_{\Sigma_\mathrm{q}}} = \varepsilon$ 且 $\mathrm{r} \triangleright \mathrm{q}!m' \leq w{\Downarrow_{\Sigma_\mathrm{r}}}$，则扩展 $w\cdot \mathrm{q} \triangleleft \mathrm{p}?m$ 不符合信道要求，那么协议 S 就满足广义接收一致性。

我们强调它与点对点的接收一致性有两处不同： [54]: w 必须是协议在 s_1' 处重新初始化后的协议前缀，而不是 s_2'；并且我们还要求一个额外的合取条件，即 $\mathrm{r} \triangleright \mathrm{q}!m'$ 是参与者 r 在 w 中的第一个事件。显然，对于点对点信道符合性，这两个 RHS 等价，因为从 r 到 q 的消息与来自其他发送方的消息不共用信道，因此可以始终将未匹配的发送事件前置到轨迹中，而不会影响信道符合性。然而，将这一事实作为假设会预先排除 mb 和 monob 网络架构。因此，广义接收一致性的 RHS 表明 w 中存在一条从 r 到 q 的发送，但不限定它位于 w 中的什么位置；我们改为对信道符合性提出以下假设，并称之为历史无关性：对于所有 $w \in \Sigma^*_{async}, \alpha, \beta \in \Gamma^*_{sync}$， $\mathrm{p} \neq \mathrm{q} \in \mathrm{P}, m \in \mathrm{V}$，则

$$(\forall \mathrm{p} \in \mathcal{P}.\ w{\Downarrow_{\Sigma_\mathrm{p}}} \leq \mathtt{split}(\alpha\beta){\Downarrow_{\Sigma_\mathrm{p}}}) \wedge w{\Downarrow_{\Sigma_\mathrm{q}}} = \mathtt{split}(\alpha){\Downarrow_{\Sigma_\mathrm{q}}} \wedge w\cdot \mathrm{q} \triangleleft \mathrm{p}?m \text{ is channel-compliant}$$
$$\implies \exists w'.(\forall \mathrm{p} \in \mathcal{P}.\ w'{\Downarrow_{\Sigma_\mathrm{p}}} \leq \mathtt{split}(\beta){\Downarrow_{\Sigma_\mathrm{q}}}) \wedge w'{\Downarrow_{\Sigma_\mathrm{q}}} = \varepsilon \wedge w'\cdot \mathrm{q} \triangleleft \mathrm{p}?m \text{ is channel-compliant} \quad \text{(F6)}$$

直观而言，F6 允许与某条同步轨迹相合规的异步轨迹 $\alpha\beta$有选择地忘记 α中的事件，从而得到一条轨迹 w'，它与 β相合规，并且如果q 在 α中 w完成了所有事件，那么在 w中仍可接收的所有消息，在 w'中仍然可接收。因此，F6 足以确立广义接收一致性的可靠性：对于一条可通过某个接收事件扩展的执行轨迹，我们论证，该接收事件必定由迄今为止该轨迹的所有合规协议运行唯一确定；否则，我们就利用F6 构造一条与广义接收一致性相矛盾的新轨迹。

前缀可扩展性。仅对点对点网络而言，三个一致性条件已不足以保证可实现性，即使加入广义接收一致性也不够。事实证明，还存在第四种不可实现性的来源；它在点对点网络中从未出现，却会出现在其他架构中。我们用一个新的第四个一致性条件来刻画这种不可实现性来源，并将其命名为前缀可扩展性。

为说明这一新条件的必要性，考虑一个简单的直线全局规约，它仅由同步词 $\mathrm{p}_1 \rightarrow \mathrm{q} : m\cdot \mathrm{p}_2 \rightarrow \mathrm{q} : m$ 构成。尽管这个规约很简单，却不存在能够实现它的无死锁 mb（或 monob）CLTS。任何候选 CLTS 都必须具有以下死锁迹： $\mathrm{p}_2 \triangleright \mathrm{q}!m\cdot \mathrm{p}_1 \triangleright \mathrm{q}!m$。由于网络可以将 p_2 的发送事件重排到 p_1 的发送事件之前，而 q 的局部动作却要求它以相反的顺序接收，因此 CLTS 若要避免死锁，唯一的办法就是让 T_q 允许局部迹 $\mathrm{q} \triangleleft \mathrm{p}_2?m\cdot \mathrm{q} \triangleleft \mathrm{p}_1?m$。

为了排除此类不可实现性情形，前缀可扩展性要求协议前缀在按参与者相等下封闭。

定义 *5.3*（前缀可扩展性）。 协议 S 满足前缀可扩展性，当对于每个协议轨迹 $\rho \in \mathrm{pref}(L(S))$ 以及 $w \in \Sigma^*_{async}$ 使得 w 是 $\mathbb{A}$-信道合规且与ρ一致时，存在 $u \in \Sigma^*_{async}$ 使得 wu 是信道合规的，并且 $wu \equiv_{\mathcal{P}}$ 拆分(ρ)。

请注意，上述协议违反了前缀可扩展性：令 $\rho = \mathsf{p}_1 \to \mathsf{q} : m \cdot \mathsf{p}_2 \to \mathsf{q} : m$和$w = \mathsf{p}_2 \triangleright \mathsf{q}!m$，那么 w与 ρ一致，且是 多播 信道合规的。然而， w所有与 拆分(ρ) 按参与者分别相等的扩展，均不满足多播信道合规性，因为它们会发生死锁。

广义一致性条件。 因此，我们对可实现性的广义刻画定义为四个条件的合取。

定义 *5.4*（广义一致性条件）。 当协议 S 满足发送一致性、广义接收一致性、无混合选择和前缀可扩展性时，称其在网络架构 $\mathbb{A}$ 下满足广义一致性条件。

我们介绍为证明精确性而对网络架构施加的其余三个假设。我们将 F1 至 F6 这六个假设统称为信道合规事实；这些假设共同构成充分条件，使我们的网络参数化刻画可靠且完备。令 $w \in \Sigma^*_{async}$ 为信道合规的。

F2 对于所有 $\mathsf{p} \neq \mathsf{q} \in \mathcal{P}$和 $m \in \mathcal{V}$， $|w\Downarrow_{\mathsf{p}\triangleright\mathsf{q}!m}| \geq |w\Downarrow_{\mathsf{q}\triangleleft\mathsf{p}?m}|$。 F4 对于所有 $\rho \in \Gamma^*_{sync}$，如果 $w \equiv_{\mathcal{P}}$ 拆分(ρ)，那么 w 是信道匹配的。F5 对于所有 $x \in \Sigma_!$， $y \in \Sigma_?$，如果 wy 是信道合规的，那么 wxy 是信道合规的。

这些基本假设贯穿整个证明。例如，F2 表明消息不能在被发送之前被接收。

下面的定理指出，假设网络架构满足我们的信道合规事实，那么我们的广义一致性条件相对于可实现性是可靠且完备的。定理 5.5 的证明已在 Rocq 中完全机械化，并以针对点对点 点对点 网络的 Rocq 机械化工作为基础，见 [57]。

定理 5.5（广义一致性条件的精确性）。令 $\mathbb{A}$ 为满足 *F1* 至 *F6* 的网络架构，并令 $\mathcal{S}$ 为全局协议。那么，$\mathcal{S}$ 在$\mathbb{A}$ 下可实现，当且仅当它在 $\mathbb{A}$下满足广义一致性条件。

6 将信道合规事实归约为缓冲区公理

有了网络参数化可实现性刻画和精确性证明之后，剩下的就是确定给定的网络架构 $\mathbb{A}$ 是否适合作为该框架的实例。根据迄今为止的技术发展，这相当于证明 $\mathbb{A}$-信道符合性满足定理所假设的六项信道合规事实。然而，若要针对每一种网络架构分别证明这些事实，即使情况最好也会十分繁琐且重复，最坏时则并非易事：尤其是历史无关性（F6）要求对一个词中彼此既不按参与者也不按信道因果有序的事件进行全局推理。作为证明信道合规事实所需工作量的实证，针对 F1至F6 的 同步广播 Rocq 形式化证明竟包含约 5000 行代码！

因此，我们进一步将信道合规事实归约为一组简单、操作性的缓冲区数据结构规范，这些规范只描述 $\mathbb{A}$ 中每个单独信道上的操作行为。我们将其称为缓冲区公理，而这组缓冲区公理构成了网络架构的公理化模型，我们的可实现性刻画适用于该模型。

下文中，对于网络架构 $\mathbb{A}$，我们用 O 表示针对 $x \in$ M 的所有操作$ins(x), rem(x) : \mathrm{B} \rightharpoonup \mathrm{B}$ 的集合。稍微滥用一下记法，我们将有限序列

$\delta = o_1 \dots o_n \in \mathrm{O}^*$ 与通过按反向顺序复合 $o_1, \dots, o_n$ 得到的偏函数 $o_n \circ \cdots \circ o_1$ 等同起来，并将空序列 ε 与 B 上的恒等函数等同起来。换言之，δ 被视为复合操作，用以表示 δ 的各个组成操作所产生的累积效应。我们进一步用 $f \bullet x$ 表示反向函数应用，即 $x \bullet f = f(x)$。

定义 *6.1*（公理化网络模型）。我们用 $\mathfrak{A}$ 表示满足以下性质的所有网络架构的集合 $\mathbb{A}$，对于任意 $b \in \mathrm{B}$、x、$y \in \mathrm{M}$ 和 $\delta \in \mathrm{O}^*$:

B1 $b \bullet \mathrm{ins}(x)$ 已定义。B2 $b_0 \bullet \mathrm{ins}(x) \bullet \mathrm{rem}(x)$ 已定义。B3 $b_0 \bullet \mathrm{rem}(x)$ 未定义。B4 若 $b \bullet \mathrm{rem}(x)$ 已定义，则 $b \bullet \mathrm{ins}(y) \bullet \mathrm{rem}(x)$ 已定义。B5 若 $b \bullet \mathrm{ins}(x) \bullet \delta$已定义，且 $\mathrm{rem}(x)$ 不出现在 δ 中，则 $b \bullet \delta$已定义。B6 若 $b \bullet \mathrm{rem}(x)$ 已定义，则 $b \bullet \mathrm{ins}(x) \bullet \mathrm{rem}(x) = b \bullet \mathrm{rem}(x) \bullet \mathrm{ins}(x)$。B7 若 $x \neq y$，则 $b \bullet \mathrm{ins}(x) \bullet \mathrm{rem}(y) = b \bullet \mathrm{rem}(y) \bullet \mathrm{ins}(x)$。B8 若 $b \bullet \mathrm{rem}(x)$ 未定义，$b \bullet \mathrm{ins}(x) \bullet \delta \bullet \mathrm{rem}(x)$ 已定义，且 $\mathrm{rem}(x)$ 不出现在δ中，则 $b \bullet \mathrm{ins}(x) \bullet \delta \bullet \mathrm{rem}(x) = b \bullet \delta$。

直观地说，B1 表明 *ins* 是全的，B2 表明人们总是可以向空信道插入并立即移除同一条消息，B3 表明不能从空信道中移除，B4 表明增加插入不会使后续移除失效，B5 表明未匹配的插入可以省略，B6 表明在该消息此前已被插入的前提下，移除与同一消息的插入左交换，B7 表明移除与不同消息的插入左交换，B8 表明相互匹配的插入与移除彼此抵消。

现在很容易看出，先进先出队列和多重集满足这些公理。因此，例 4.2 中的所有网络架构都属于 $\mathfrak{A}$。然而，一些常见的信道数据结构被排除在外。特别是，B1 排除了有界信道，B8排除了允许消息重复的信道。

下述引理表明，我们的网络模型蕴含定理 5.5 的前提条件。

引理 6.2. 所有 $\mathbb{A} \in \mathfrak{A}$ 均满足 *F1* 至 *F6*。

推论 6.3. 设 $\mathbb{A} \in \mathfrak{A}$。则对所有协议 $\mathcal{S}$，$\mathcal{S}$ 在 $\mathbb{A}$ 下可实现，当且仅当它满足广义一致性条件。

引理 6.2 的证明见 §A.1。例如，为证明 F3 成立，可证明更强的性质：拆分(α) 依 α的长度作归纳法可证其为信道匹配。在归纳步骤中，利用 B2 可证明扩展后的字仍保持信道合规，然后利用 B8 可证明它使所有信道为空，即信道匹配。

7 广义一致性条件的衍生

本节将基于我们对网络无关可实现性的可靠且完备刻画，推导一系列结果。首先，我们介绍具有依赖细化的符号协议。遵循 [54]，中的蓝图，我们证明广义 一致性条件 同样可以编码为定义在符号协议转移关系上的最小与最大不动点。这使我们能够定义一种可靠且相对完备的算法，用于判定示例 4.2 中考虑的五种网络架构下符号协议的可实现性。作为一段插曲，我们利用这种编码，确立每种网络架构下可实现协议类之间的关系。最后，我们给出所考虑架构下有限片段的复杂性结果。

7.1 检查符号协议的可实现性

符号协议 [54] 定义在一个固定但未指定的一阶消息值背景理论之上（例如线性整数算术）。我们假定一阶公式具有标准的语法和语义，并用 $\mathcal{F}$ 表示自由变量取自无限集合 X 的一阶公式集合。我们假定这些变量的取值范围为消息值集合 $\mathcal{V}$。

定义 *7.1*（符号协议 *[54]*）。符号协议是一个元组 $\mathbb{S} = (S, R, \Delta, s_0, \rho_0, F)$ 其中

- S 是一个有限的控制状态集，
- R 是一个有限的寄存器变量集，
- $\Delta \subseteq S \times \mathcal{P} \times X \times \mathcal{P} \times \mathcal{F} \times S$ 是由以下形式的符号转换构成的有限集合 $s \xrightarrow{\mathsf{p} \to \mathsf{q}:x\{\varphi\}} s'$ 其中公式 φ 具有自由变量 $R \uplus R' \uplus \{x\}$，用于表示关联新旧寄存器值（R 和 R'）以及发送值 x 的转换约束，
- $s_0 \in S$ 是初始控制状态，
- $\rho_0 \colon R \to \mathrm{V}$ 是初始寄存器赋值，并且
- $F \subseteq S$ 是终态。

在本节余下部分，我们固定一个符号协议 $\mathbb{S} = (S, R, \Delta, s_0, \rho_0, F)$，其具体化满足 GCLTS 假设。

首先，我们展示如何将广义接收一致性（GRC）编码为 μCLP。我们用 $\mathrm{RHS} := w \in \mathrm{pref}(\mathrm{L}(\mathrm{S}^{s_i})) \wedge$ 表示 GRC 蕴含式右侧的四个合取项 $w{\Downarrow_{\Sigma_\mathsf{q}}} = \varepsilon \wedge \mathsf{r} \triangleright \mathsf{q}!m' \leq w{\Downarrow_{\Sigma_\mathsf{r}}} \wedge w \cdot \mathsf{q} \triangleleft \mathsf{p}?m$ 是信道合规的。令 $s_1 \mathsf{p} \xrightarrow{\to\ m} s_2 \mathsf{q}:, s_1' \mathsf{r} \xrightarrow{\to\ m'} s_2' \in T_{\mathsf{q}:}$，使得 s_1, s_2 可由 q 同时到达。我们首先对其前提进行分类讨论，将 GRC 分解为两个条件。令 GRC(a) 表示 $(\mathsf{r} \neq \mathsf{p} \implies \neg\, \exists\, w.\, \mathrm{RHS}(w))$，令 GRC(b) 表示 $(\mathsf{r} = \mathsf{p} \wedge m' \neq m \implies \neg\, \exists\, w.\, \mathrm{RHS}(w))$。显然，对于采用 FIFO 信道的网络模型，GRC(b) 不可满足。在 FIFO 信道中，接收顺序遵循发送顺序。根据全局协议的语义，每个参与者的事件都具有全序，因此 $\mathsf{p} \triangleright \mathsf{q}!m'$ 由 p 执行的发送先于 $\mathsf{p} \triangleright \mathsf{q}!m$ 由 p 执行的发送；后者必定存在于 w 中，因为信道合规的词满足先发送后接收的顺序。 m 能够被 q 在 w 中接收的唯一情况是 q 已经接收了 m'，但合取项 $w{\Downarrow_{\Sigma_\mathsf{q}}} = \varepsilon$ 指出，q 在 w中没有任何事件。总之，只需针对 $\mathbb{A}$ =包检查 GRC(b)。

接下来，我们展示如何将 $\mathrm{RHS}(w)$ 编码为最小不动点。Li 等人 [54] 定义了一族谓词 $\mathrm{avail}_{\mathsf{p},\mathsf{q},\mathcal{B}}(x_1, s_2, \boldsymbol{r}_2)$，用于刻画 x_1 是否可作为点对点网络中从 q 发往 p 的第一条消息而可用，同时使用 B 跟踪因果依赖。B 跟踪因发送动作因果依赖于 q 首先从 r 接收而被阻止发送消息的参与者集合。然而，他们的 avail 谓词是特定于点对点网络架构的。

我们给出可用性的网络参数化版本，它参数化于所选发送方p 和接收方q （我们正在寻找它们之间的消息交换）、用于跟踪参与者之间因果依赖关系的阻塞集合$\mathcal{B}$，以及网络架构 $\mathbb{A} \in$ {点对点、同步广播、多播、monob、包}。

定义 *7.2*（符号可用性）。

$$\begin{aligned}\mathsf{avail}_{\mathbb{A},\mathsf{p},\mathsf{q},\mathcal{B}}(x_1,s,\boldsymbol{r}) \coloneqq_\mu \quad & \Big(\bigvee_{\substack{(s,\mathsf{r}\to\mathsf{t}:x\{\varphi\},s')\in\Delta\\ \mathsf{r}\in\mathcal{B}}} \exists x\,\boldsymbol{r'}.\,\mathsf{avail}_{\mathbb{A},\mathsf{p},\mathsf{q},\mathcal{B}\cup\{\mathsf{t}\}}(x_1,s',\boldsymbol{r'})\wedge\varphi\Big)\\ \vee\ & \Big(\bigvee_{\substack{(s,\mathsf{r}\to\mathsf{t}:x\{\varphi\},s')\in\Delta\\ \mathsf{r}\notin\mathcal{B}\\ \mathsf{ch}(\mathsf{r},\mathsf{t})\neq\mathsf{ch}(\mathsf{p},\mathsf{q})\vee\mathsf{t}\notin\mathcal{B}\vee\mathbb{A}=\mathsf{bag}}} \exists x\,\boldsymbol{r'}.\,\mathsf{avail}_{\mathbb{A},\mathsf{p},\mathsf{q},\mathcal{B}}(x_1,s',\boldsymbol{r'})\wedge\varphi\Big)\\ \vee\ & \Big(\bigvee_{\substack{(s,\mathsf{p}\to\mathsf{q}:x\{\varphi\},s')\in\Delta\\ \mathsf{p}\notin\mathcal{B}}} \varphi[x_1/x]\Big)\ .\end{aligned}$$

定义中的最后一个析取项处理以下情况：消息 x_1 从p 发送后，可在符号状态q 中立即由$(s,\boldsymbol{r})$接收，且 p 未被阻止发送。第一个析取项跳过发送方被阻止的所有迁移，并在对 $\mathcal{B}$ 递归调用 avail时将接收方加入其中。第二个析取项跳过发送方未被阻止、但消息交换不会干扰消息的迁移 x_1 从p 到q。也就是说，以下三种情形之一成立：消息交换发生在不同的信道上；消息交换发生在同一信道上，但接收方未被阻止；或者 $\mathbb{A}$ 是包（即， x 和 x_1 可以重新排序）。

支撑我们广义avail 谓词的关键观察是：决定如何处理协议运行期间遇到的每个事件时，各个析取项的条件都可以完全参数化为通信拓扑。为此，请注意，当我们将ch(r,t) ≠ ch(p,q) 实例化为$\mathsf{r}\neq\mathsf{p}\vee\mathsf{t}\neq\mathsf{q}$，并将 $\mathbb{A}$ = 包实例化为 $\bot$ 用于点对点箱式网络，同时移除 q 始终是$\mathcal{B}$的元素这一假设后，我们便得到 [54]中的定义5.5。

以下等价关系展示了如何实例化 可用性 ，以判定每种齐次网络架构下的广义接收一致性。注意，RHS 将以下参数作为隐式参数p、q、r m， m' 和 s_1'。

命题 7.3。对于 $\mathbb{A}\in\{\text{p2p, bag}\}$， $\exists\, w.\,\mathrm{RHS}_{\mathbb{A}}(w) \Longleftrightarrow \exists\, x_1, s_2', \boldsymbol{r}.$可用性$_{\mathbb{A},\mathsf{p},\mathsf{q},\{\mathsf{q}\}}(x_1,s_2',\boldsymbol{r})$。对于$\mathbb{A}\in\{\text{sb, monob, mb}\}$， $\exists\, w.\,\mathrm{RHS}_{\mathbb{A}}(w) \Longleftrightarrow \exists\, x_1, s_2', \boldsymbol{r}.$可用性$_{\mathbb{A},\mathsf{p},\mathsf{q},\{\mathsf{q},\mathsf{r}\}}(x_1,s_2',\boldsymbol{r})$。

利用上述等价关系和我们对可用性的定义，随后可以将广义接收一致性定义为符号协议上的谓词。我们以 $\mathbb{A}$ =sb 为例给出该谓词。

定义 *7.4*（sb的符号广义接收一致性）。符号协议 $\mathbb{S}$满足 sb 的符号广义接收一致性，当且仅当对于每一对迁移 $s_1 \xrightarrow{\mathsf{p}\to\mathsf{q}:\ x_1\{\varphi_1\}} s_1' \in\Delta$ 且 $s_2 \xrightarrow{\mathsf{r}\to\mathsf{q}:\ x_2\{\varphi_2\}} s_2' \in\Delta$ 满足 $\mathsf{p}\neq\mathsf{r}$：

$$\text{产品反应}\ \mathrm{h}_{\mathsf{q}}(s_1,\ \boldsymbol{r}_1,\ s_2,\ \boldsymbol{r}_2)\ \wedge\ \varphi_1[\boldsymbol{r}_1\boldsymbol{r}_1'/\boldsymbol{r}\boldsymbol{r}']\ \wedge\ \varphi_2[\boldsymbol{r}_2\boldsymbol{r}_2'/\boldsymbol{r}\boldsymbol{r}']\ \wedge\text{可用性}_{\mathsf{sb},\ \mathsf{p},\ \mathsf{q},\ \{\mathsf{q},\mathsf{r}\}}(x_1,\ s_2',\ \boldsymbol{r}_2')\ \Longrightarrow\ \bot\ ,$$

最后，我们描述如何检查符号前缀可扩展性。首先，我们证明，对于点对点、包和同步广播网络，前缀可扩展性的一种更强形式成立。

引理 7.5。 对于 $\mathbb{A}\in\{$点对点,同步广播, 包$\}$，对于满足以下条件的每个 $\rho\in\Gamma^*_{sync}$, $w\in\Sigma^*_{async}$：w是 $\mathbb{A}$-信道合规的，并且与 ρ一致，存在 $u\in\Sigma^*_{async}$使得 wu 是信道合规的，且 $wu\equiv_{\mathcal{P}}$拆分(ρ)。

证明见 §A.2。前缀可扩展性约束全局协议的运行，并要求每个部分完成某次运行的信道合规词都可以扩展为完成某次运行。引理 7.7 表明，对于上述网络架构，每个部分完成任意同步词的信道合规词都可以扩展为完成同一个同步词。显然，引理 7.7 蕴含前缀可扩展性。使引理 7.7 得以成立的关键事实在于

能够将具有活动参与者 z 的第一个事件 p 快进到任意词 u_1 之前，同时保持信道合规，条件是 u_1 不包含以 p 为活动参与者的事件。§5 中给出的示例为 mb 和monob 信道合规性下引理 7.7 提供了简单的反例。因此，对于这两个网络，我们必须明确检查所考虑协议的运行是否具有前缀可扩展性。

为了说明前缀可扩展性的编码，我们首先注意到，任何符合 monob 或 mb 信道要求的词也符合点对点信道要求。因此，给定一个符合 monob 或 mb 信道要求的词 w，令 u 为其补全，使得 wu 符合点对点信道要求。相反，并非每个符合点对点信道要求的词都符合 monob 或 mb 信道要求。我们关注这样的事件：monob 或 mb 网络可以对其排序，而点对点网络不能。monob 和 mb 网络都可以对发送到同一个 接收方的独立发送排序，其总顺序由 ρ 固定。因此，u 的构造在遇到此类独立发送时会失败。单箱网络还可以对发送到不同接收方的独立发送排序。在点对点网络中，只有发送到不同接收方且不存在因果依赖的独立发送才能重新排序。这里的一个关键观察是，要在发送到不同接收方的独立发送之间引入因果依赖，必然需要对发送到同一接收方的独立发送排序。由此我们得出结论：monob 和 mb 的前缀可扩展性是等价的。

因此，要同时检查多播与 monob 的前缀可扩展性，必须验证：当一条消息被发送时，不存在另一条发送给同一接收方且可用的不同消息。所需的递归谓词在本质上类似于各种网络模型的 avail 谓词。形式上，只需确保从任意转移 $s_1 \mathsf{p}{\to}\mathsf{q}{:}x\{\varphi\}\!\longrightarrow s_2 \in \Delta$ 出发，不存在一条经过 s_2、随后又经过转移 $s_3 \mathsf{r}{\to}\mathsf{q}{:}x\{\varphi\}\!\longrightarrow s_4$ 的运行，使得 $\mathsf{r} \neq \mathsf{p}$ 的发送不依赖于 p 或 q。我们将此刻画为谓词 avail′。

定义 *7.6*（符号可用性）。

$$\begin{aligned}\mathsf{avail}'_{\mathsf{q},\mathcal{B}}(s,\boldsymbol{r}) :=_\mu \quad & \Big(\bigvee_{\substack{(s,\mathsf{r}\to\mathsf{t}:x\{\varphi\},s')\in\Delta\\ \mathsf{r}\in\mathcal{B}}} \exists x\,\boldsymbol{r}'.\,\mathsf{avail}'_{\mathsf{q},\mathcal{B}\cup\{\mathsf{t}\}}(s',\boldsymbol{r}')\wedge\varphi\Big)\\ \vee \Big(&\bigvee_{\substack{(s,\mathsf{r}\to\mathsf{t}:x\{\varphi\},s')\in\Delta\\ \mathsf{r}\notin\mathcal{B}\\ \mathsf{t}\neq\mathsf{q}\vee\mathsf{t}\notin\mathcal{B}}} \exists x\,\boldsymbol{r}'.\,\mathsf{avail}'_{\mathsf{q},\mathcal{B}}(s',\boldsymbol{r}')\wedge\varphi\Big) \;\vee\; \Big(\bigvee_{\substack{(s,\mathsf{r}\to\mathsf{q}:x\{\varphi\},s')\in\Delta\\ \mathsf{r}\notin\mathcal{B}}} \exists x\,\boldsymbol{r}'.\,\varphi\Big)\ .\end{aligned}$$

我们还需要一个谓词，用以刻画到达给定控制状态的所有可达寄存器与变量赋值，这些赋值来自 [56]。

定义 *7.7*（符号协议中的可达性）。设 $s \in S_\circ$。则，

$$\mathsf{reach}(s',\boldsymbol{r}') :=_\mu \quad (s' = s_0 \wedge \boldsymbol{r}' = \rho_0) \;\vee\; \Big(\bigvee_{(s,\mathsf{p}\to\mathsf{q}:x\{\varphi\},s')\in\Delta} \exists x\,\boldsymbol{r}.\,\mathsf{reach}(s,r)\wedge\varphi\Big)\ .$$

基于此，我们提出适用于 $\mathbb{A} \in \{\mathsf{mb}、\mathsf{monob}\}$ 的符号前缀可扩展性。

定义 *7.8*（邮箱、单一邮箱的前缀可扩展性）。一个符号协议 $\mathbb{S}$ 满足 $\mathbb{A}$-前缀可扩展性，适用于 $\mathbb{A} \in \{$多播、monob$\}$ 当且仅当对于每个转移 $s_{\mathsf{p}\to\mathsf{q}:x\{\varphi\}}\!\longrightarrow s' \in \Delta$:

$$\mathsf{reach}(s',\boldsymbol{r}') \;\wedge\; \mathsf{avail}'_{\mathsf{q},\{\mathsf{p},\mathsf{q}\}}(s',\boldsymbol{r}') \implies \bot\ .$$

7.2 可实现性关系

根据可用性的网络参数化定义，显然 sb-可用性蕴含 p2p-可用性，而 p2p-可用性又蕴含 bag-可用性。由于可用性在广义接收一致性中处于否定位置，且另外两个一致性条件是网络无关的，因此我们得到以下结论。

引理 7.9（可实现性关系）。任何可由包实现的全局协议都可由点对点实现，而任何可由点对点实现的全局协议都可由同步广播实现。

这些包含关系的严格性由图 G_a 和G_b 图 5 以及图 6中的示例见证。由 多播 与 monob 前缀可扩展性等价这一事实，可得以下结论：

推论 7.10。 一个全局协议是多播-可实现的，当且仅当它是monob-可实现的。

我们进一步猜想，多播/monob 类包含于点对点类之中。

我们注意到，由引理 同步广播、点对点 和包 网络架构所诱导的关系与其语义之间的关系一致；其语义定义为消息序列图集合或可执行轨迹集合 [13, 34]。我们将在第 9 节进一步详细讨论这一点。

7.3 有限片段的复杂度

我们将 co-NP 完全性复杂度结果从 [54] 推广到其他网络架构，并在附录 §A.3中给出详细讨论。我们还表明，co-NP 困难性下界并不需要 GCLTS 的完全一般性；特别地，该归约即使对于带有定向选择的全局类型 [42]也成立。

定理 7.11. 对于 $\mathbb{A} \in$ {同步广播，多播，monob，包}，有限协议的可实现性是 *co-NP* 完全的。

推论 7.12. 对于 $\mathbb{A} \in$ {点对点，同步广播，多播，monob，包}，带有定向选择的全局多方会话类型的可实现性是 *co-NP* 完全的。

8 实现与评估

我们扩展了Sprout [56] 协议可实现性检查器与验证工具，使其以网络架构为参数。最初的Sprout 以符号化全局协议 $\mathbb{S}$（如§7.1 所述）作为输入。随后，它检查 $\mathbb{S}$ 是否表示一个 GCLTS，若是，则检查其是否可实现。可实现性检查将 [54] 的一致性条件编码为 μCLP 实例。MuVal 被用作所生成 muCLP 实例的后端求解器。Sprout 还为验证协议特定属性提供了初步支持。

我们对 Sprout 的扩展用我们新的网络参数化广义 一致性条件 编码取代了原来的 [54] 编码，具体讨论见 §一致性条件。我们按照 §7.1中的讨论，将该编码实例化为以下五种具体的网络架构 $\mathbb{A} \in$ {点对点、同步广播、多播、monob、包}。新工具 Sprout($\mathbb{A}$)允许用户在调用工具时从这些架构中选择一种。

在我们未发现任何实现缺陷的前提下，Sprout($\mathbb{A}$) 相对于 MuVal 的可靠性和完备性而言是可靠且完备的。由于符号协议的可实现性不可判定，MuVal 在某些生成的 μCLP 实例上可能会发散。

我们对 Sprout ($\mathbb{A}$) 的评估旨在回答以下问题：
Q1 Sprout ($\mathbb{A}$) 能否正确判定可实现性？Q2 Sprout($\mathbb{A}$) 的性能（即验证时间／终止行为）在不同网络架构下会如何变化？

基准测试。为帮助回答这两个问题，我们从 Sprout [56]评估中使用的基准测试套件入手。这些基准测试涵盖了多方会话类型文献中的各种基准测试套件，同时还纳入了一些新的基准测试以进一步丰富多样性，尤其是增加有趣的不可实现基准测试数量。示例包括各种 Web 服务和分布式协议的理想化规约。为进一步帮助我们回答 Q1 ，我们添加了一些精选的微型基准测试，使每一对所考察的网络架构在可实现性方面都能区分开来。这些基准测试将在 §7.2 中讨论。

源示例		$\lvert\mathcal{P}\rvert$	p2p	Time	sb	Time	mb	Time	monob 时间		bag	Time
[83]	计算器	2	✓	0.6s	✓	0.7s	✓	0.7s	✓	0.8s	×	16.5秒
	斐波那契	2	✓	0.5s	✓	0.5s	✓	0.5s	✓	0.5s	×	3.3s
	HigherLower	3	✓	15.8秒	✓	13.6秒	×	17.0秒	×	18.5秒	×	45.6秒
	HTTP	2	✓	0.5s	✓	0.4s	✓	0.4s	✓	0.5s	×	24.4秒
	协商	2	✓	1.0s	✓	1.0s	✓	1.0s	✓	1.0s	×	26.1秒
	在线钱包	3	✓	17.5秒	✓	16.6秒	✓	16.5秒	✓	20.6秒	×	100.0秒
	SH	3	✓	237.1秒	✓	244.0秒	✓	256.1秒	✓	257.2秒	?	T/O
[79]	机票	2	✓	0.6s	✓	0.6s	✓	0.6s	✓	0.6s	×	3.8s
	旅行社	2	✓	9.6s	✓	9.7s	×	12.2秒	×	11.6秒	×	18.3秒
	双买家	3	✓	4.2s	✓	4.0s	×	6.3s	×	6.3s	✓	9.3s
	双缓冲	3	✓	1.5s	✓	1.5s	×	2.4s	×	2.3s	✓	2.5s
	OAuth	3	✓	6.5s	✓	6.6s	✓	10.5秒	✓	10.3秒	×	18.1秒
	加减	3	✓	5.5s	✓	5.3s	×	7.7s	×	8.6s	✓	13.2秒
	RingMax	7	✓	3.7s	✓	3.6s	✓	5.3s	✓	6.8s	✓	16.5秒
	SimpleAuth	2	✓	0.5s	✓	0.5s	✓	0.5s	✓	0.5s	×	4.7s
	TravelAgency2	2	✓	0.5s	✓	0.5s	✓	0.5s	✓	0.5s	✓	3.7s
[53]	发送有效性-是	4	✓	1.9s	✓	2.7s	✓	2.7s	✓	2.7s	✓	3.6s
	发送有效性-否	4	×	1.9s	×	1.9s	×	2.7s	×	2.7s	×	3.6s
	接收有效性-是	3	✓	5.3s	✓	5.3s	✓	6.5s	✓	5.9s	✓	7.8s
	接收有效性-否	3	×	3.7s	×	3.8s	×	4.6s	×	4.0s	×	5.4s
[54]	符号化双投标人-是	3	✓	24.0秒	✓	21.6秒	✓	25.2秒	✓	22.0秒	✓	31.7秒
	符号化双投标人-否1	3	×	23.9秒	×	20.3秒	×	22.9秒	×	18.3秒	×	29.5秒
	图12-是	3	✓	2.0s	✓	2.0s	✓	8.7s	✓	8.5s	✓	12.8秒
	图12-否	3	×	3.0s	×	3.0s	×	9.7s	×	10.7秒	×	13.8秒
	符号发送有效性-是	4	✓	6.6s	✓	6.5s	✓	10.0秒	✓	10.5秒	✓	16.0s
	符号发送有效性-否	4	×	5.6s	×	5.4s	×	8.3s	×	8.8s	×	13.2s
	符号接收有效性-是	3	✓	6.4s	✓	6.3s	×	8.7s	×	8.4s	✓	11.8s
	符号接收有效性-否	3	×	8.0s	×	7.8s	×	12.3秒	×	12.3秒	×	14.5秒
[?]	前向认证-是	3	✓	10.5秒	✓	10.5秒	✓	15.6秒	✓	16.1秒	×	27.3秒
	前向认证-否	3	?	T/O	?	T/O	×	157.4秒	×	163.0秒	×	176.1秒
[56]	符号化双竞标者无2	3	×	23.8秒	×	29.5秒	×	31.1秒	×	28.5秒	×	40.1秒
	更高-更低-终极	3	✓	13.3秒	✓	13.1秒	×	20.5秒	×	25.3秒	×	30.8秒
	高低胜出	3	?	T/O	✓	36.3秒	×	37.5秒	×	45.1秒	✓	62.2秒
	高低未胜	3	×	13.4s	×	12.9s	×	20.9s	×	29.9s	×	45.3s
	高低加密-是	4	✓	10.0秒	✓	9.8s	×	12.9秒	×	14.3秒	✓	25.0秒
	高低加密-否	4	×	77.3秒	×	88.8秒	×	100.0秒	×	109.2秒	×	126.1秒
	高低混合	3	×	33.6秒	×	33.8秒	×	36.4秒	×	36.2秒	×	41.3秒
new	包袋-无点对点-是	2	✓	0.4s	✓	0.4s	✓	3.4s	✓	0.6s	×	1.8s
	MB-无点对点-是	3	✓	0.9s	✓	0.9s	×	1.3s	×	1.3s	✓	1.3s
	点对点-否-同步广播-是	4	×	4.0s	✓	4.1s	×	5.3s	×	5.3s	×	6.7s
	多播-否-monob-否	3	×	3.2s	✓	3.2s	×	4.1s	×	4.1s	×	5.4s
	monob-否-多播-否	4	✓	1.3s	✓	1.3s	×	1.9s	×	1.9s	✓	1.9s
	平均值			6.7s		5.8s		7.6s		7.1s		12.6秒

表 1. 五种网络架构的运行时间和验证结果。对于每个示例，我们报告参与者的数量（$\lvert\mathcal{P}\rvert$），以及每种架构的结果：✓表示可实现，×表示不可实现，?表示因超时而无法得出结论；同时报告运行时间，每个 μCLP 实例的超时限为 15 秒（超时）。最后一行所列所有基准测试的平均运行时间，仅根据未发生任何配置超时的基准测试计算。

实验与结果。为回答这两个问题，我们在完整的基准测试套件和所有考察的网络架构上运行 Sprout($\mathbb{A}$)。实验在一台配备 Apple M3 芯片和 16GB 内存的 2024 款 MacBook Air 上进行，运行时间取 10 次运行的平均值。结果汇总于表 1。报告的运行时间为 GCLTS 检查时间与可实现性检查时间之和，各个 μCLP 实例的超时时间另行列出。

正确性（*Q1*）。我们首先观察到，点对点盒的可实现性结果与 [56] 中报告的通用基准测试结果一致。其次，请注意，这些结果也与引理7.9中确定的网络架构之间的可实现性关系一致。例如，在同步广播 下被确定为可实现的基准测试子集，严格包含点对点下的相应子集。最后，我们对分离小型基准测试中的各个结果以及随机选取的其他结果进行了人工检查。在所有情况下，我们都认定生成的结果是正确的。

性能（*Q2*）。Sprout（𝔸）显示出与先前支持的点对点相比，新网络架构具有相近的运行时间和终止行为，其中包 的表现最慢。包 表现较慢是意料之中的，因为消息缓冲区中更宽松的排序会扩大消息可用性检查需要探索的状态空间，而该检查用于实现 GRC 条件。总体而言，平均运行时间之间的关系似乎与 §7.2中确定的网络架构之间的理论关系相吻合。

9 相关工作

全局协议规范。 以消息序列图形式呈现的全局协议规范早在1993年便被国际电信联盟标准采用 [46]，随后于2005年纳入 UML [70]，并成为 Web 服务编排描述语言的一部分 [80]。学术界广泛研究全局规范，其形式包括多方会话类型和编排式编程。多方会话类型（多方会话类型）已在多种编程语言中得到广泛实现，包括 Python [20, 65, 67], 、Java [43, 44], 、C [68], 、Go[9, 51], 、Scala [10], 、Rust [17, 49], 、OCaml [45] 和 F# [66]。多方会话类型的应用领域包括操作系统 [22], 、高性能计算 [18, 41, 69], 、信息物理系统 [59, 60], 和 Web 服务 [82]。编舞式编程框架已在 Java [32],、Haskell [74], 和 Rust [48, 52] 中实现，并应用于分布式架构 [71], 、密码安全协议 [25], 和信息物理系统 [15]。关于多方会话类型和编舞适用性的全面综述，请参阅 [81] 和 [62] 。

并发理论中的网络参数化结果。 Bollig 等人 [6] 和 Giusto 等人 [34] 研究可同步性问题，该问题关注能否根据通信有限状态机的“同步”执行对其进行精确近似。Bollig 等人 [6] 提出了一个基于 MSO 逻辑和特殊树宽的框架，andusesit 以证明关于存在性和全称性 k-有界性 [27], 弱同步性以及弱 k-同步性 [7] 的点对点和 mb 通信的可判定性结果。

Giusto 等人 [34] 将 Bollig 等人 [6] 的研究推广至另外五种通信模型，这些模型定义为消息序列图（MSC）的集合：同步广播、monob、包、因果有序，以及可通过同步通信实现。MSC 是由异步事件构成的偏序图，为每个参与者指定一个全序。因此，MSC 可以表示未匹配发送和乱序接收，但不能表示分支行为或递归。Giusto 等人 [34] 进一步将可同步性推广至任何能够用特殊树宽有界的 MSC 类来表达的概念。因此，这一推广建立在将通信模型和可同步性属性编码为 MSO 公式，以及 Courcelle 定理之上；该定理适用于任何特殊树宽有界的图。

非 FIFO（等价于袋式）通信与 FIFO 通信一并在[58], 中进行了研究，该研究探讨了相对于通信有限状态机的 HMSC 安全可实现性问题。HMSC 在两个维度上推广了 GCLTS：首先，虽然规范必须满足 FIFO 限制，但不要求是 1-同步的；其次，分支选择不受限制，从而放宽了 GCLTS 的发送方驱动选择假设。与我们的设定不同，HMSC 没有采用无限词语义。Lohrey [58] 通过利用 FIFO 通信与非 FIFO 通信之间的归约来转移安全可实现性的上界，证明所有（不可）判定性结果都可推广到袋式通信。

近期，文献 [35]研究了点对点通信和会合同步通信中全局类型的可实现性。与我们的信道符合性概念类似，该工作将通信模型定义为一组线性化。会合同步本质上是全局信道界限为 1 的异步通信，类似于 文献 [13] 和 [34], 中的 rsc；在我们的设定中，它对应于拆分 (ρ)异步字。 [35]则取消了对图结构的所有限制，

尤其允许发送方驱动选择。无论采用特定于网络还是网络无关的方式，都没有针对有限全局类型可实现性的判定程序。相反，点对点 FIFO 可实现的全局类型与同步可实现的全局类型之间的关系，是通过直接定义可实现性来确立的。

Chevrou 等人 [13] 研究了 [34], 所考察的同样七种通信模型，并建立了一个层级，其对 [34] 同步广播 和 多播通信的排序与 [13] 不同：前者认为两者不可比较，而 [34] 则认为同步广播 包含 多播。这种差异源于全称量词与存在量词的不同：Chevrou 等人将通信模型定义为线性化集合，其中所有线性化都遵循该通信模型；而 Giusto 等人将通信模型定义为 MSC 集合，其中只需有一个 MSC 存在线性化遵循该通信模型。他们的重点是建立分类体系，而非针对一种或多种通信模型的具体判定问题。他们的结果已在 TLA 中形式化+。

我们的网络架构概念将每个信道视为一个单独的消息缓冲区，这在全局协议规范框架中很常见。Engels 等人 [21] 考虑了一种更细粒度的模型，其中每个信道都关联一个输入缓冲区和一个输出缓冲区。这样一来，每次消息交换都会多出第三种异步事件，对应于将消息从输出缓冲区传送到输入缓冲区。随后，他们研究了诸多问题，包括不同通信拓扑下先进先出信道上的 MSC 可实现性问题。然而，他们对可实现性的定义与我们的不同：他们只检查 MSC 的所有线性化是否都符合网络语义，却不检查该集合能否由一个 CLTS 精确实现，且其中的参与者只能进行局部观测。他们的结果也不容易推广到混合消息序列图（HMCSs），因此排除了规范中的递归和循环等特性。

我们的公理化网络模型允许网络架构在同一个网络中采用语义各异的信道，例如先进先出和包。Clemente 等人 [14] 研究了此类混合网络中有限状态 CLTS 的可达性问题的可判定性。具体而言，他们完整刻画了能够保持可判定性的通信拓扑。

工具。 据我们所知，Sprout(𝔸) 是首个能够针对多种不同网络架构检查全局协议可实现性的工具。在本文中，我们以 Sprout [56],为基础；该工具实现了 [54] 中针对点对点情形提出的可实现性检查符号化算法。Sprout 最接近的竞争工具是 Session* [83] 以及结合精化的 Rumpsteak [79]。详细比较请参阅 [56] 。

同步性。 本文的研究未涉及同步通信模型。针对同步通信模型的自顶向下验证已作为独立研究方向得到广泛探讨，尤其体现在同步多方会话类型 [11, 31, 72, 77]中。Zielonka 自动机 [84] 及其“现实”变体 [1] ，是构建统一理论、研究具有同步或异步语义的协议的自然起点。值得注意的是， [54] 和本文采用的发送一致性条件，可以视为第三个语义条件的一个特例；该条件为因果闭合 [1,定义 9（LC3）],，用于刻画全局规范的同步可实现性。这一联系表明，有望实现一种可同时处理同步和异步通信的可实现性刻画。

10 结论

我们提出了一个网络参数化的、精确的全局协议可实现性解决方案，以及一个简单的公理化网络模型，用于刻画该解决方案所适用的网络架构。借助这一解决方案，我们还推导出了有限状态全局协议片段以及通常所考虑的网络架构的可判定性和复杂性结果

适用于可能具有无限状态空间和数据域的全局协议的符号算法。我们将后者实现于一个现有工具中，并展示了由此产生的新工具 Sprout(𝔸) 能够在不牺牲性能的情况下，扩大跨网络架构的适用范围。

我们的工作带来的一个启示是，在可实现性问题中，所选的网络架构不会影响综合：对于所有考察的网络架构，当且仅当通过局部投影得到的规范实现能够实现该全局协议时，该全局协议才是可实现的。这种模块化特性让我们希望，我们的结果可以直接用于拓展自顶向下方法的适用范围。

致谢。 本研究获得美国国家科学基金会资助，资助协议编号为 2304758。

参考文献

[1] S. Akshay、Ionut Dinca、Blaise Genest 和 Alin Stefanescu。2013。现实异步自动机的实现。载于 IARCS 软件技术与理论计算机科学基础年会，FSTTCS 2013，2013年12月12日至14日，印度古瓦哈提（LIPIcs，第24卷），Anil Seth 和 Nisheeth K. Vishnoi（编）。Schloss Dagstuhl - Leibniz-Zentrum für Informatik，213–224。doi:10.4230/LIPICS.FSTTCS.2013.213[2] Rajeev Alur、Kousha Etessami 和 Mihalis Yannakakis。2003。消息序列图推断。《IEEE 软件工程汇刊》 29, 7 2003 623 633 10 1109 2003 1214326 ( ), –。doi: . /TSE. 。Rajeev Alur 和 Mihalis Yannakakis。 。消[3] 息序列图模型检测。载于 CONCUR '99：1999 并发理论，第10届国际会议，荷兰埃因霍温，1999年8月24日至27日，论文集（计算机科学讲义，第 *1664* 114 129 10 1007 3 540 48320 9_10 卷），Jos C. M. Baeten[4] 和 Sjouke Mauw（编）。施普林格，–。doi: . / - - - 2012 Laura Bocchi、Romain Demangeon 和 Nobuko Yoshida。 。多方多会话逻辑。载于可信全球计算——第7届国际研讨会，TGC 2012，英国泰恩河畔纽卡斯尔，2012年9月7日至8日，修订版精选论文（计算机科学讲义，第8191卷），Catuscia Palamidessi 和 Mark Dermot Ryan（编）。施普林格， 97 111 10 1007 978 3 642 41157 1_7 –。doi: . / - - - - Laura Bocchi、Kohei Honda、Emilio Tuosto 和 Nobuko[5] Yoshida。2010。分布式多方交互的契约式设计理论。载于 CONCUR 2010——并发理论，第21届国际会议，CONCUR 2010，法国巴黎，2010年8月31日至9月3日，论文集（计算机科学讲义，第6269卷），Paul Gastin 和 François Laroussinie（编）。施普林格， 162 176 10 1007 978 3 642 15375 4_12 –。doi: . / - - - - Benedikt Bollig、Cinzia Di Giusto、Alain[6] Finkel、Laetitia Laversa、Étienne Lozes 和 Amrita Suresh。2021。可同步性判定的统一框架。载于第32届并发理论国际会议，CONCUR 2021，2021年8月24日至27日，线上会议（LIPIcs，第203卷），Serge Haddad 和 Daniele Varacca（编）。Schloss Dagstuhl - Leibniz-Zentrum für Informatik，14:1–14:18。doi:10.4230/LIPICS.CONCUR.2021.14[7] Ahmed Bouajjani、Constantin Enea、Kailiang Ji 和 Shaz Qadeer。2018。关于有界异步条件下消息传递程序验证的完备性。载于计算机辅助验证——第30届国际会议，CAV 2018，作为联合逻辑会议 FloC 2018 的一部分举行，英国牛津，2018年7月14日至17日，论文集，第二部分（计算机科学讲义，第10982卷），Hana Chockler 和 Georg Weissenbacher（编）。施普林格，372–391。doi:10.1007/978-3-319-96142-2_23[8] Daniel Brand 和 Pitro Zafiropulo。1983。关于通信有限状态机。《ACM期刊》30，2（1983），323–342。doi:10.1145/322374.322380[9] David Castro-Perez、Raymond Hu、Sung-Shik Jongmans、Nicholas Ng 和 Nobuko Yoshida。2019。使用 Go 中的角色参数化会话类型进行分布式编程：面向动态实例化通信结构的静态类型端点 API。《ACM程序设计语言汇刊》 3, 2019 29 1 29 30 10 1145 3290342 POPL（），：–：。doi: . / David[10] Castro-Perez 和 Nobuko Yoshida。 。可动态更新的多方会话协议：从无界协议生成并发 Go 代码。载于第37届欧洲面向对象编程会议，ECOOP 2023，2023年7月17日至21日，美国华盛顿州西雅图（LIPIcs，第263卷），Karim Ali 和 Guido Salvaneschi （编）。Schloss Dagstuhl - Leibniz-Zentrum für Informatik， 2023 6 1 6 30 10 4230 2023 6：–：。doi: . /LIPICS.ECOOP. 。[11] Tzu-Chun Chen、Mariangiola Dezani-Ciancaglini 和 Nobuko Yoshida。2024 。关于会话类型中子类型的精确性：十年之后。载于第26届声明式编程原理与实践国际研讨会论文集，PPDP 2024，意大利米兰，2024年9月9日至11日，Alessandro Bruni、Alberto Momigliano、Matteo Pradella、Matteo Rossi 和 James Cheney（编）。ACM， 2024 1 2 3 10 1145 3678232 3678258：–：。doi: . / 。Florent Chevrou、Aurélie[12] Hurault、Shin Nakajima 和 Philippe Quéinnec。2019 。异步通信模型图谱。载于形式化方法。FM 2019 国际研讨会——葡萄牙波尔图，2019年10月7日至11日，修订版精选论文，第二部分（计算机科学讲义，第12233卷），Emil Sekerinski、Nelma Moreira、José N. Oliveira、Daniel Ratiu、Riccardo Guidotti、Marie Farrell、Matt Luckcuck、Diego Marmsoler、José Creissac Campos、Troy Astarte、Laure Gonnord、Antonio Cerone、Luis Couto、Brijesh Dongol、Martin Kutrib、Pedro Monteiro 和 David Delmas（编）。施普林格， 307 322 10 1007 978 3 030 54997 8_20 –。doi: . / - - - - Florent Chevrou、Aurélie Hurault 和 Philippe Quéinnec[13]。2016。异步通信的多样性。《形式化方法》28，5（2016），847–879。doi:10.1007/S00165-016-0379-X [14] Lorenzo Clemente、Frédéric Herbreteau 和 Grégoire Sutre。2014。采用先进先出和袋式通道的通信自动机的可判定拓扑。载于 CONCUR 2014——并发理论——第25届国际会议，CONCUR 2014，意大利罗马，2014年9月 *2 5, 2014* 8704 281 296 10 1007 978 3 662 44584 6_20 –。论文集，第 卷。施普林[15] 格，–。doi: . / - - - - Luís Cruz-Filipe 和 2016 Fabrizio Montesi。 。实践中的编排。载于分布式对象、组件与系统的形式化技术——第36届 IFIP WG 6.1 国际会议，FORTE 2016，作为第11届分布式计算技术国际联合会议 DisCoTec 2016 的一部分举行，希腊克里特岛伊拉克利翁，2016年6月6日至9日，论文集（计算机科学讲义，第9688卷），Elvira Albert 和 Ivan Lanese（编）。施普林格，114–123。doi:10.1007/978-3-319-39570-8 8_

[16] Luís Cruz-Filipe 和 Fabrizio Montesi。2020年。编舞式编程的核心模型。《理论计算机科学》802（ 2020 38 66 10 1016 2019 07 005），–。doi：。/J.TCS.。。Zak Cutner、Nobuko Yoshida 和 Martin Vassor。[17] 2022年。使用多方会话类型在 Rust 中实现无死锁异步消息重排。载于 PPoPP '22：第27届 ACM SIGPLAN并行编程原理与实践研讨会，韩国首尔，2022年4月2日至6日，Jaejin Lee、Kunal Agrawal 和 Michael F. Spear（编）。ACM，246–261。doi:10.1145/3503221.3508404[18] Jan de Muijnck-Hughes 和 Wim Vanderbauwhede。2019年。硬件接口的类型规则。载于第33届欧洲面向对象编程会议（ECOOP 2019）（莱布尼茨国际信息学论文集（LIPIcs），第134卷），Alastair F. Donaldson（编）。Schloss Dagstuhl – Leibniz-Zentrum für Informatik，德国达格施图尔，6:1–6:27。doi:10.4230/LIPIcs.ECOOP.2019.6[19] Romain Delpy、Anca Muscholl 和 Grégoire Sutre。2024年。基于自动机的可同步邮箱通信方法。载于第35届国际并发理论会议 CONCUR 2024，2024年9月9日至13日，加拿大卡尔加里（LIPIcs，第311卷），Rupak Majumdar 和 Alexandra Silva（编）。Schloss Dagstuhl - Leibniz信息学中心，22:1–22:19。doi:10.4230/LIPICS.CONCUR.2024.22[20] Romain Demangeon、Kohei Honda、Raymond Hu、Rumyana Neykova 和 Nobuko Yoshida。2015年。实用的可中断对话：使用多方会话类型和 Python 进行分布式动态验证。《形式化方法与系统设计》46，3（2015），197–225。doi:10.1007/S10703-014-0218-8[21] André Engels、Sjouke Mauw 和 Michel A. Reniers。2002年。消息序列图通信模型的层次结构。《科学计算程序设计》 44, 3 2002 253 292 10 1016 0167 6423 02 00022 9（），–。doi：。/S - ( ) - Manuel [22] Fähndrich、Mark Aiken、Chris Hawblitzel、Orion Hodson、Galen C. Hunt、James R. Larus 和 Steven Levi。2006年。Singularity操作系统中对快速可靠的基于消息通信的语言支持。载于2006年欧洲系统会议论文集，2006年4月18日至21日，比利时鲁汶，Yolande Berbers 和 Willy Zwaenepoel（编）。ACM，177–190。doi:10.1145/1217935.1217953[23] Alain Finkel 和 Étienne Lozes。2023年。通信有限状态机的可同步性不可判定。《逻辑方法与计算机科学》 19, 4 2023 10 46298 19 4 33 2023（）。doi：。/LMCS-（：）Robert G. Gallager、[24] Pierre A. Humblet 和 Philip M. Spira。。最小权重生成树的1983分布式算法。《ACM程序设计语言汇刊》 5, 1 1983 66 77 10 1145 357195 357200（），–。doi：。/。Joshua [25] Gancher、Sydney Gibson、Pratap Singh、Samvid Dharanikota 和 Bryan Parno。2023。。Owl：通过信息流类型系统对安全协议进行组合式验证。载于第44届 IEEE安全与隐私研讨会，SP *2023, 21 25, 2023* 1130 1147 10 1109 46215 2023 10179477 美国加利福尼亚州旧金山，5月-。IEEE，–。doi：。/SP。。Thomas Gazagnaire、Blaise Genest、Loïc Hélouët、P. S. Thiagarajan 和 Shaofa Yang。。因果消息序列图。载于 CONCUR 2007——并发理论，第18届国际会议 CONCUR 2007，葡萄牙里斯本，2007年9月3日至8日，论文集（《计算机科学讲义》，第4703卷），Luís Caires 和 Vasco Thudichum Vasconcelos（编）。施普林格， 166 180 10 1007 978 3 540 74407 8_12 –。doi：。/- - - - Blaise Genest、Dietrich Kuske 和 Anca Muscholl[27]。。克莱尼定理及存在界限通信自动机的模型检测2006算法。《信息与计算》 204, 6 2006 920 956 10 1016 2006 01 005（），–。doi：。/ [28] J.IC.。。Blaise Genest 和 Anca Muscholl2005。。消息序列图：综述。载于第五届并发在系统设计中应用国际会议（ACSD 2005），2005年6月6日至9日，法国圣马洛。IEEE计算机学会，2–4。doi:10.1109/ACSD.2005.25[29] Blaise Genest、Anca Muscholl 和 Doron A. Peled。2003年。消息序列图。载于《并发与佩特里网讲座：佩特里网进展》 *[*本教程卷源自2003年9月在德国艾希施泰特举行的第4届佩特里网高级课程（ACPN 2003）。除在 ACPN 2003 上所作的讲座外，还委托撰写了其他章节*]*（《计算机科学讲义》，第3098卷），Jörg Desel、Wolfgang Reisig 和 Grzegorz Rozenberg（编）。施普林格， 537 558 10 1007 978 3 540 27755 2_15 –。doi：。/- - - - Blaise Genest、Anca Muscholl、[30] Helmut Seidl 和 Marc Zeitoun。2006年。无限状态高级消息序列图：模型检测与可实现性。《计算机与系统科学杂志》 72, 4 2006 617 647 10 1016 2005 09 007（），–。doi：。/J.JCSS.。。Silvia [31] Ghilezan、Svetlana Jaksic、Jovanka Pantovic、Alceste Scalas 和 Nobuko Yoshida。2019年。同步多方会话的精确子类型。《逻辑与代数方法程序设计》104（2019），127–173。doi:10.1016/J.JLAMP.2018. 12.002[32] Saverio Giallorenzo、Fabrizio Montesi 和 Marco Peressotti。2024年。Choral：面向对象的编舞式编程。《ACM程序设计语言汇刊》 46, 1 2024 1 1 1 59 10 1145 3632398（），：–：。doi：。/ Saverio [33] Giallorenzo、Fabrizio Montesi、Marco Peressotti、David Richter、Guido Salvaneschi 和 Pascal Weisenburger。2021年。多方语言：编舞式与多层式案例（Pearl）。载于第35届欧洲面向对象编程会议（ECOOP 2021），2021年7月11日至17日，丹麦奥胡斯（线上会议）（LIPIcs，第194卷），Anders Møller 和 Manu Sridharan（编）。Schloss Dagstuhl - Leibniz信息学中心，22:1–22:27。doi:10.4230/LIPICS. ECOOP.2021.22

[34] Cinzia Di Giusto、Davide Ferré、Laetitia Laversa 和 Étienne Lozes。2023年。消息传递通信模型的偏序视角。《ACM程序设计语言汇刊》 7, 2023 1601 1627 10 1145 3571248 POPL（），–。doi：。/ Cinzia Di Giusto、Étienne Lozes 和 Pascal Urso。2025。。多方会话类型的可实现性与可补全性。CoRR abs/ 2507 17354 2025 2507 17354 10 48550 2507 17354 。（）。arXiv：。doi：。/ARXIV.。Jonas Kastberg Hinrichsen、Jesper Bengtson 和 Robbert Krebbers。2020。。Actris：分离逻辑中基于会话类型的推理。《ACM程序设计语言汇刊》 4, 2020 6 1 6 30 10 1145 3371074 POPL（），：–：。doi：。/ Jonas Kastberg Hinrichsen、Jesper Bengtson 和 Robbert Krebbers。2022。。Actris .：分离逻辑中基于异步会话类型的推理。《逻辑方法与计算机科学》 18, 2 2022 10 46298 18 2 16 2022 （）。doi：。/ LMCS-（：）Jonas Kastberg Hinrichsen、Jules Jacobs 和 Robbert Krebbers。2024。。Multris：分离逻辑中多方消息传递的功能验证。（2024）。https://jihgfee.github.io/papers/multris_manuscript.pdf [39] Andrew K. Hirsch 和 Deepak Garg。2021年。Pirouette：高阶类型化函数式编排。CoRR abs/ 2111 03484 2021 2111 03484 2111 03484 。（）。arXiv：。https://arxiv.org/abs/。Andrew K. Hirsch 和 Deepak Garg。2022。。Pirouette：高阶类型化函数式编排。《ACM程序设计语言汇刊》 6, 2022 1 27 10 1145 3498684 POPL（），–。doi：。/ Kohei Honda、Eduardo R. B. Marques、Francisco Martins、Nicholas Ng、Vasco Thudichum Vasconcelos 和 Nobuko Yoshida。2012年。使用会话类型验证 MPI程序。载于《消息传递接口最新进展——第19届欧洲 MPI 用户组会议 EuroMPI 2012》，奥地利维也纳，2012年9月23日至26日，论文集（《计算机科学讲义》，第7490卷），Jesper Larsson Träff、Siegfried Benkner 和 Jack J. Dongarra（编）。施普林格，291–293。doi:10.1007/978-3-642-33518-1_37[42] Kohei Honda、Nobuko Yoshida 和 Marco Carbone。2008年。多方异步会话类型。载于第35届 ACM SIGPLAN-SIGACT编程语言原理研讨会论文集，POPL 2008，美国加利福尼亚州旧金山，1月 7 12, 2008, 273 284 10 1145 1328438 1328472 - George C. Necula 和 Philip Wadler （编）。ACM，–。doi：。/。Raymond Hu 和 Nobuko Yoshida。2016。。通过端点 API 生成进行混合会话验证。载于《软件工程基础方法——第19届国际会议 FASE 2016》，作为欧洲软件理论与实践联合会议 ETAPS 2016 的一部分举行，荷兰埃因霍温，2016年4月2日至8日，论文集（《计算机科学讲义》，第9633卷），Perdita Stevens 和 Andrzej Wasowski（编）。施普林格，401–418。doi:10.1007/978-3-662-49665-7 24_[44] Raymond Hu 和 Nobuko Yoshida。2017年。多方会话类型中的显式连接动作。载于《软件工程基础方法——第20届国际会议 FASE 2017》，作为欧洲软件理论与实践联合会议 ETAPS 2017 的一部分举行，瑞典乌普萨拉，2017年4月22日至29日，论文集（《计算机科学讲义》，第10202卷），Marieke Huisman 和 Julia Rubin（编）。施普林格，116–133。doi:10.1007/978-3- 662-54494-5 7_[45] Keigo Imai、Rumyana Neykova、Nobuko Yoshida 和 Shoji Yuen。2020年。使用全局协议组合子的多方会话编程。载于34 17，2020，2020，15 166 第欧洲面向对象编程会议，ECOOP 11月-柏林，德国（线上会议）（LIPIcs，第卷），Robert Hirschfeld 和 Tobias Pape（编）。Schloss Dagstuhl - Leibniz信息学中心，9:1–9:30。doi:10.4230/LIPICS.ECOOP.2020.9[46] 国际电信联盟。2011年。ITU-T建议Z.120：消息序列图（MSC）。ITU-T建议Z.120。国际电信联盟，日内瓦。https://www.itu.int/rec/T-REC-Z.120- 201102-I/en[47] Jules Jacobs、Jonas Kastberg Hinrichsen 和 Robbert Krebbers。2023年。分离逻辑中的依赖会话协议：从第一原理出发（函数式珍珠）。《ACM程序设计语言汇刊》 7, 2023 768 795 10 1145 3607856 ICFP（），–。doi：。/ Shun Kashiwa、Gan Shen、Soroush Zare 和 Lindsey Kuper。2023。。可移植、高效且实用的库级编舞式编程。arXiv: 2311 11472 [ ] 2311 11472 . cs.PL https://arxiv.org/abs/。Nicolas Lagaillardie、Rumyana Neykova 和 Nobuko Yoshida。2022。。安然应对恐慌：使用多方会话类型进行仿射Rust编程（工件）。《达格施图尔工件系列》 8, 2 2022 09 1 09 16 10 4230 8 2 9 （），：–：。doi：。/DARTS.。[50]。Leslie Lamport。2019。。分布式系统中的时间、时钟与事件排序。载于《并发：Leslie Lamport作品集》，Dahlia Malkhi（编）。ACM， 179 196 10 1145 3335772 3335934 –。doi：。/。Julien Lange、Nicholas Ng、Bernardo Toninho 和 Nobuko Yoshida。2018。。使用行为类型对 Go 中的消息传递进行静态验证的框架。载于第40届国际软件工程会议 ICSE 2018 论文集，瑞典哥德堡，2018年5月27日至6月3日，Michel Chaudron、Ivica Crnkovic、Marsha Chechik 和 Mark Harman（编）。ACM，1137–1148。doi:10.1145/3180155.3180157[52] 加州大学圣克鲁兹分校语言、系统与数据实验室。[无日期]。ChoRus：Rust中的编舞式编程。https://lsd- ucsc.github.io/ChoRus/introduction.html。访问日期：2025-07-10。[53] Elaine Li、Felix Stutz、Thomas Wies 和 Damien Zufferey。2023年。使用自动机实现完备多方会话类型投影。载于计算机辅助验证——第35届国际会议 CAV 2023，法国巴黎，2023年7月17日至22日，论文集第三部分（《计算机科学讲义》，第13966卷），Constantin Enea 和 Akash Lal（编）。施普林格，350–373。doi:10.1007/978-3-031-37709-9

[54] Elaine Li、Felix Stutz、Thomas Wies 和 Damien Zufferey。2025年。《刻画具有无限状态与数据的全局协议的可实现性》。PACMPL 9，《面向对象编程、系统、语言与应用》（OOPSLA）（2025年）。[55] Elaine Li、Felix Stutz、Thomas Wies 和 Damien Zufferey。2025年。《刻画具有无限状态与数据的全局协议的可实现性》。arXiv: 2411 05722 [ ] 2411 05722 。cs.PL https://arxiv.org/abs/ 。Elaine Li、[56] 2025 Felix Stutz、Thomas Wies 和 Damien Zufferey。。《Sprout：一种符号化多方协议验证器》。载于《计算机辅助验证——第37届国际会议，CAV 2025》，克罗地亚萨格勒布，2025年7月23日至25日，论文集，第三部分（《计算机科学讲义》，第15933卷），Ruzica Piskac 和 Zvonimir Rakamaric（编）。施普林格，304–317。doi:10.1007/978-3-031-98682-6 16_[57] Elaine Li 和 Thomas Wies。2025年。《经认证的全局多方协议可实现性》。载于《第16届交互式定理证明国际会议，ITP 2025》，2025年9月28日至10月1日，冰岛雷克雅未克（LIPIcs，第352卷），Yannick Forster 和 Chantal Keller（编）。Schloss Dagstuhl - Leibniz-Zentrum für Informatik，15:1–15:20。doi:10.4230/ LIPICS.ITP.2025.15[58] Markus Lohrey。2003 309, 1 3 2003 529 554 10 1016 2003 08 。高级消息序列图的可实现性：弥合差距。《理论计算机科学》- ( )，– 。doi: 。/J.TCS. . . 402 [59] Rupak Majumdar、Marcus Pirron、Nobuko Yoshida 和 Damien Zufferey。2019年。《用于机器人交互的运动会话类型》（前沿创意论文）。载于《第33届欧洲面向对象编程会议，ECOOP 2019》，2019年7月15日至19日，英国伦敦（LIPIcs，第134卷），Alastair F. Donaldson（编）。Schloss Dagstuhl - Leibniz-Zentrum für Informatik，28:1–28:27。doi:10.4230/LIPICS.ECOOP.2019.28[60] Rupak Majumdar、Nobuko Yoshida 和 Damien Zufferey。2020年。《多方运动协调：从编排到机器人程序》。《ACM程序设计语言汇刊》 4, 2020 134 1 134 30 10 1145 3428202 OOPSLA ( )，：– : 。doi: 。/ [61] 1997 Sjouke Mauw 和 Michel A. Reniers。.《高级消息序列图》。载于《SDL ' 测试的时刻：SDL、MSC 与趋势——第8届国际SDL论坛》，法国埃夫里，1997年9月23日至29日，论文集，Ana R. Cavalli 和 Amardeo Sarma（编）。Elsevier，291–306。 97 [62] 2023 10 1017 9781108981491 Fabrizio Montesi。。[63] 2002 《编排导论》。剑桥大学出版社。doi: 。/ Rémi Morin。。《消息序列图的可识别集合》。载于《STACS 第 届理论计算机科学专题讨论会》，法国昂蒂布-儒安雷潘，2002年3月14日至16日，论文集（《计算机科学讲义》，第 2002, 19 2285 523 534 10 1007 3 540 45841 7_43 卷），Helmut Alt 和 [64] 1999 Afonso Ferreira（编）。施普林格，– 。doi: 。/ - - - Anca Muscholl 和 Doron A. Peled。。《消息序列图与 Mazurkiewicz轨迹上的判定问题》。载于《计算机科学数学基础 1999》，第24届国际研讨会，MFCS'99，波兰斯克拉尔斯卡-波伦巴，1999年9月6日至10日，论文集（《计算机科学讲义》，第1672卷），Miroslaw Kutylowski、Leszek Pacholski 和 Tomasz Wierzbicki（编）。施普林格， 81 91 10 1007 3 540 48340 3_8 – 。doi: 。/ - - - Rumyana Neykova、Laura Bocchi 和 Nobuko Yoshida。。《多方对话的定时运行时监控》。《计算机形式化方面》 [65] 2017 29, 5 2017 877 910 10 1007 00165 017 0420 8 ( )，– 。doi: 。/S - - - Rumyana Neykova、[66] 2018 Raymond Hu、Nobuko Yoshida 和 Fahd Abdeljallal。。《会话类型提供程序：在 F# 中为带精化的分布式协议生成编译时 API》。载于《第27届编译器构造国际会议论文集，CC 2018》，2018年2月24日至25日，奥地利维也纳，Christophe Dubach 和 Jingling Xue（编）。ACM，128–138。doi:10.1145/3178372.3179495[67] Rumyana Neykova 和 Nobuko Yoshida。2017年。《多方会话参与者》。《计算机科学逻辑方法》13, 1（2017年）。doi:10.23638/LMCS-13(1:17)2017[68] Nicholas Ng、Nobuko Yoshida 和 Kohei Honda。2012年。《多方会话 C：借助消息优化实现安全并行编程》。载于《对象、模型、组件、模式——第50届国际会议，TOOLS 2012》，捷克共和国布拉格，2012年5月29日至31日，论文集（《计算机科学讲义》，第7304卷），Carlo A. Furia 和 Sebastian Nanz（编）。施普林格， 202 218 10 1007 978 3 642 30561 0_15 – 。doi: 。/ - - - - Xinyu Niu、[69] Nicholas Ng、Tomofumi Yuki、Shaojun Wang、Nobuko Yoshida 和 Wayne Luk。2016年。《EURECA 编译：周期可重构电路的自动优化》。载于《第26届现场可编程逻辑与应用国际会议，FPL 2016》，瑞士洛桑，2016年8月29日至9月2日，Paolo Ienne、Walid A. Najjar、Jason Helge Anderson、Philip Brisk 和 Walter Stechele（编）。IEEE， 1 4 10 1109 2016 7577359 – 。doi: 。/FPL. 。对象管理组。[70] [ ] n. d.。《统一建模语言（UML）网站》。https://www.uml.org/。访问日期：2025-07-10。[71] Giuseppe De Palma、Saverio Giallorenzo、Jacopo Mauro、Matteo Trentin 和 Gianluigi Zavattaro。2024年。《迈向函数即服务编舞编程语言：示例与应用》。arXiv:2406.09099 [cs.PL] https://arxiv.org/abs/ 2406 09099 。Kirstin Peters 和 Nobuko Yoshida。2024年。[72] 《混合选择多方会话中的分离与可编码性》。载于《第39届ACM/IEEE计算机科学中的逻辑年度研讨会论文集，LICS 2024》，爱沙尼亚塔林，2024年7月8日至11日，Pawel Sobocinski、Ugo Dal Lago 和 Javier Esparza（编）。ACM，62:1–62:15。doi:10.1145/3661814.3662085

[73] Abhik Roychoudhury、Ankit Goel 和 Bikram Sengupta。2012年。《符号消息序列图》。《ACM软件工程与方法汇刊》 21, 2 2012 12 1 12 44 10 1145 2089116 2089122 ( ), : – : 。doi: 。/ 。Gan Shen、Shun Kashiwa 和 Lindsey Kuper。2023。《HasChor：面向所有人的函数式编舞编程》（函数式珍品）。CoRR abs/2303.00924（2023年）。arXiv:2303.00924 doi:10.48550/ARXIV.2303.00924[75] Felix Stutz。2024年。《异步通信协议的可实现性——选择的力量》。博士论文。德国凯撒斯劳滕工业大学。https://kluedo.ub.rptu.de/frontdoor/index/index/docId/ 8077[76] Bernardo Toninho 和 Nobuko Yoshida。2017。《多方会话类型中的数据认证》。《逻辑与代数方法程序学报》 90 2017 61 83 10 1016 2016 11 005 ( ), – 。doi: 。/J.JLAMP. . . Thien Udomsrirungruang 和 Nobuko Yoshida。2025。[77]《自顶向下还是自底向上？同步多方会话类型的复杂度分析》。《ACM程序设计语言汇刊》 9, 2025 1040 1071 10 1145 3704872 POPL ( ), – 。doi: 。/ Hiroshi Unno、Tachio Terauchi、Yu Gu 和 Eric Koskinen。2023。[78]《用于时序验证的模块化原始-对偶不动点逻辑求解》。《ACM程序设计语言汇刊》 7, 2023 2111 2140 10 1145 3571265 POPL ( ), – 。doi: 。/ Martin Vassor 和 Nobuko Yoshida。2024。[79]《多方消息传递协议的精化：与规范无关的理论与实现》。载于《第38届欧洲面向对象编程会议，ECOOP 2024》，2024年9月16日至20日，奥地利维也纳（LIPIcs，第313卷），Jonathan Aldrich 和 Guido Salvaneschi（编）。Schloss Dagstuhl - Leibniz-Zentrum für Informatik，41:1–41:29。doi:10.4230/LIPICS.ECOOP.2024.41[80] Web服务编舞工作组。2005年。《Web服务编舞描述语言 1.0版》。W3C候选推荐标准。万维网联盟（W3C）。可从 http://www.w3.org/TR/2005/CR-ws- cdl-10-20051109/ 获取。[81] Nobuko Yoshida。2024年。《基于多方会话类型的编程语言实现》。载于《主动对象语言：当前研究趋势》，Frank S. de Boer、Ferruccio Damiani、Reiner Hähnle、Einar Broch Johnsen 和 Eduard Kamburjan（编）。《计算机科学讲义》，第14360卷。施普林格，147–165。doi:10.1007/978-3-031- 51060-1 6_[82] Nobuko Yoshida、Raymond Hu、Rumyana Neykova 和 Nicholas Ng。2013年。《Scribble协议语言》。载于《可信全球计算——第8届国际研讨会，TGC 2013》，阿根廷布宜诺斯艾利斯，2013年8月30日至31日，修订精选论文（《计算机科学讲义》，第8358卷），Martín Abadi 和 Alberto Lluch-Lafuente（编）。施普林格， 22 41 10 1007 978 3 319 05119 2_3 – 。doi: 。/ - - - - Fangyi Zhou、Francisco Ferreira、Raymond Hu、Rumyana Neykova 和 Nobuko Yoshida。2020年。[83]《面向多方协议的静态验证精化》。《ACM程序设计语言汇刊》 4, 2020 148 1 148 30 10 1145 3428216 OOPSLA ( ), : – : 。doi: 。/ Wieslaw Zielonka。1987。[84]《有限异步自动机札记》。《RAIRO理论信息与应用》( ), – 。doi:10.1051/ITA/1987210200991 21, 2 1987 99 135

A 附录

A.1 §的证明6

在本节中，我们证明引理 6.2。设 $\mathbb{A} \in \mathfrak{A}$ 是一个满足我们的公理化网络模型的网络架构。我们需要证明 $\mathbb{A}$ 满足 F1 至 F6。为方便起见，我们在此处集中重述这些事实：

定义 *A.1*（信道合规事实）。 设 $w \in \Sigma^*_{async}$ 符合信道合规性。

F1 对所有 $x \in \Sigma_!, wx$，均符合信道合规性。 F2 对所有 $\mathrm{p} \neq \mathrm{q} \in \mathrm{P}$ 和 $m \in \mathrm{V}$，$|w\Downarrow_{\mathrm{p}\triangleright\mathrm{q}!m}| \geq |w\Downarrow_{\mathrm{q}\triangleleft\mathrm{p}?m}|$。 F3 对所有 $\rho \in \Gamma^*_{sync}$，$\mathrm{split}(\rho)$ 均符合信道合规性。 F4 对所有 $\rho \in \Gamma^*_{sync}$，若 $w \equiv_{\mathcal{P}} \mathrm{split}(\rho)$，则 w 与信道匹配。 F5 对所有 $x \in \Sigma_!, y \in \Sigma_?$，若 wy 符合信道合规性，则 wxy 符合信道合规性。 F6 设 $\alpha, \beta \in \Gamma^*_{sync}, \mathrm{p} \neq \mathrm{q} \in \mathrm{P}$ 和 $m \in \mathrm{V}$，使得对于所有 $\mathrm{p} \in \mathrm{P}$，$w\Downarrow_{\Sigma_\mathrm{p}} \leq \mathrm{split}(\alpha\beta)\Downarrow_{\Sigma_\mathrm{p}}$，且 $w\Downarrow_{\Sigma_\mathrm{q}} = \mathrm{split}(\alpha)\Downarrow_{\Sigma_\mathrm{q}}$，并且 $w \cdot \mathrm{q} \triangleleft \mathrm{p}?m$ 符合信道合规性。那么，存在 w'，使得 w' 与 $\beta, w'\Downarrow_{\Sigma_\mathrm{q}} = \varepsilon$ 合规，且 $w' \cdot \mathrm{q} \triangleleft \mathrm{p}?m$ 符合信道合规性。

信道状态是从信道到缓冲区内容的映射，因此，当且仅当一个词在每个信道上的投影对于相应缓冲区都有定义时，该词才是信道合规的。因此，在下面的证明中，我们只需推理缓冲区插入与移除操作以及缓冲区状态。

引理 A.2（F2）。对于所有 $\mathrm{p} \neq \mathrm{q} \in \mathcal{P}$ 和 $m \in \mathcal{V}$，$|w\Downarrow_{\mathrm{p}\triangleright\mathrm{q}!m}| \geq |w\Downarrow_{\mathrm{q}\triangleleft\mathrm{p}?m}|$。

证明。对 $|w|$ 使用强归纳法。令 $w = w'x$。我们分类讨论 x 是发送事件还是接收事件。如果 $x \in \Sigma_!$，结论立即成立。如果 $x \in \Sigma_?$，令 $x = \mathrm{s} \triangleleft \mathrm{r}?m$。我们要证明 $|w'x\Downarrow_{\mathrm{r}\triangleright\mathrm{s}!m}| \geq |w'x\Downarrow_{\mathrm{s}\triangleleft\mathrm{r}?m}|$。根据归纳假设，$|w'\Downarrow_{\mathrm{r}\triangleright\mathrm{s}!m}| \geq |w'\Downarrow_{\mathrm{s}\triangleleft\mathrm{r}?m}|$。只需分析 $|w'\Downarrow_{\mathrm{r}\triangleright\mathrm{s}!m}| = |w'\Downarrow_{\mathrm{s}\triangleleft\mathrm{r}?m}|$ 的情形，即在 w' 中发送的每条消息 m 都已被接收。令 $w' = w''y$。接下来，我们分类讨论 y 是发送事件还是接收事件。如果 $y \in \Sigma_!$，并且 y 的消息不等于 m，则由 (B7) 可得 $w' \cdot \mathrm{s} \triangleleft \mathrm{r}?m \cdot y$ 令 m_y 为 y 的消息值。如果 $m_y \neq m$，那么 $w = w' \cdot \mathrm{p} \triangleright \mathrm{q}!m \cdot \mathrm{s} \triangleleft \mathrm{r}?m$，由推广到词的 B7 可知，$w'.\mathrm{s} \triangleleft \mathrm{r}?m \cdot \mathrm{p} \triangleright \mathrm{q}!m$ 到达的信道状态与 w 相同，因此我们对 $w' \cdot \mathrm{s} \triangleleft \mathrm{r}?m$ 使用归纳假设。如果 $m_y = m$，注意到由于消息带有发送方与接收方信息的标记，可得 $y = \mathrm{r} \triangleright \mathrm{s}!m$。我们直接对 w' 使用归纳假设。如果 $y \in \Sigma_?$，我们先对 $w'y$ 使用归纳假设，以找到 w' 中至少一次出现的 $\mathrm{r} \triangleright \mathrm{s}!m$。我们在 w' 中找到 $\mathrm{r} \triangleright \mathrm{s}!m$ 的首次出现：令 $w' = u_1 \cdot \mathrm{r} \triangleright \mathrm{s}!m \cdot u_2$，使得 $u_1\Downarrow_{\mathrm{r}\triangleright\mathrm{s}!m} = \varepsilon$。如果 $u_1 \cdot \mathrm{s} \triangleleft \mathrm{r}?m$ 缓冲区合规，我们就对 u_1 使用归纳假设。否则，我们在 u_2 中找到 $\mathrm{s} \triangleleft \mathrm{r}?m$ 的首次出现。如果 $\mathrm{s} \triangleleft \mathrm{r}?m$ 没有出现在 u_2 中，我们就将归纳假设应用于 w'。否则，我们进一步将 $u_2 = v_1 \cdot \mathrm{s} \triangleleft \mathrm{r}?m \cdot v_2$ 拆分，使得 $v_1\Downarrow_{\mathrm{s}\triangleleft\mathrm{r}?m} = \varepsilon$。然后，我们对 $w = u_1 \cdot \mathrm{r} \triangleright \mathrm{s}!m \cdot v_1 \cdot \mathrm{s} \triangleleft \mathrm{r}?m \cdot v_2$ 应用 (B8)，从而得出 $u_1v_1v_2$ 到达的状态与 w 相同。我们对 $u_1v_1v_2 \cdot \mathrm{q} \triangleleft \mathrm{p}?m$ 使用归纳假设来证明结论。□

引理 A.3 (F1)。 对于所有 $x \in \Sigma_!$，wx 符合信道合规要求。

证明。 由 (B1说明 insert 的全性可立即得出。 □

引理 A.4 (F5)。 对于所有 $x \in \Sigma_!$，$y \in \Sigma_?$，若 wy 符合信道合规要求，则 wxy 符合信道合规要求。

证明。 由 (B4) 可立即得出。 □

引理 A.5 (F3)。 对于所有 $\rho \in \Gamma^*_{sync}$，拆分(ρ) 符合信道合规要求。

证明。我们证明一个更强的性质，即对于所有 $w \in \Sigma^*_{async}$，$\mathrm{split}(w)$ 都能到达空信道状态 ξ_0。我们通过对 $|w|$ 进行归纳法来证明这一点。基例显然成立。在归纳

步骤中，令 $w = w' \cdot \mathrm{p} \to \mathrm{q} : m$。根据归纳假设，$\mathrm{split}(w')$ 能到达空信道状态 ξ_0。令 $c = \mathrm{ch}(\mathrm{p}, \mathrm{q})$。于是，$\xi_0(c) = b_0$。根据 (B2)，$b_0 \bullet \mathrm{ins}(x) \bullet \mathrm{rem}(x)$ 有定义。根据 (B8)，$b_0 \bullet \mathrm{ins}(x) \bullet \mathrm{rem}(x) = b_0$。因此，$\mathrm{split}(w' \cdot \mathrm{p} \to \mathrm{q} : m)$ 能到达空信道状态，故其信道合规。□

引理 A.6 (F4)。设 $\mathcal{T}_{\mathbb{A}}$ 是一个 *CLTS*，$w \in \Sigma^*_{async}$ 并且 $\rho \in \Gamma^*_{sync}$ 满足 $w \equiv_{\mathcal{P}} \text{拆分}(\rho)$，且 w 是 $\mathcal{T}_{\mathbb{A}}$ 的一条轨迹。那么，在沿 w 到达的配置中，所有信道都是空的。

证明。我们通过对 $|\rho|$ 进行归纳，证明每个信道合规的词 $\bar{w}$（它按角色等于某个 $\rho \in \Gamma^*_{sync}$ 的 $\mathrm{split}(\rho)$）都会到达空信道。在基例中，$\rho = \varepsilon$ 和 $w = \varepsilon$，结论显然成立。在归纳步骤中，设 $w = xw'$。我们对 x 是发送事件还是接收事件进行分类讨论。若 $x \in \Sigma_?$，根据假设，xw' 是信道合规的；根据信道符合性的前缀闭包，x 也是信道合规的。根据 (B3)，无法从空信道中移除内容，因此得到矛盾。设 $x \in \Sigma_!$ 为 $\mathrm{p} \triangleright \mathrm{q}!m$。由于 xw' 按角色等于 $\mathrm{split}(\rho)$，因此 w' 中至少有一次出现 $\mathrm{q} \triangleleft \mathrm{p}?m$。我们取第一次出现，即设 $w = \mathrm{p} \triangleright \mathrm{q}!m \cdot u_1 \cdot \mathrm{q} \triangleleft \mathrm{p}?m \cdot u_2$，且 $\mathrm{q} \triangleleft \mathrm{p}?m$ 不出现在 u_1 中。此外，设 $\rho = \rho_1 \cdot \mathrm{p} \to \mathrm{q} : m \cdot \rho_2$，且 $\mathrm{p} \to \mathrm{q} : m$ 不出现在 ρ_1 中。因此，u_1u_2 与 $\rho_1\rho_2$ 按角色完全相同，根据归纳假设，u_1u_2 会到达空信道。将 (B8) 提升到信道状态后，沿 u_1u_2 到达的信道状态与沿 xw' 到达的信道状态相同。由此得出，沿 $w = xw'$ 到达的信道状态为空信道。□

引理 *A.7 (F6)*。设 $w \in \Sigma^*_{async}$ 是信道合规的。设 α、$\beta \in \Gamma^*_{sync}$、$p \neq q \in P$ 和 $m \in V$ 满足：对于所有 $p \in P$，$w{\Downarrow_{\Sigma_p}} \leq \mathit{split}(\alpha\beta){\Downarrow_{\Sigma_p}}$，且 $w{\Downarrow_{\Sigma_q}} = \mathit{split}(\alpha){\Downarrow_{\Sigma_q}}$，并且 $w \cdot q \triangleleft p?m$ 是信道合规的。则存在 w'，使得 w' 与 β 合规，$w{\Downarrow_{\Sigma_q}} = \varepsilon$，并且 $w' \cdot q \triangleleft p?m$ 是信道合规的。

证明。我们分两步构造 w'。第一步，我们从 w 和 α 中移除匹配事件对。

断言 1。存在 w_1 和 α_1，使得 w_1 与 $\alpha_1\beta$ 合规，w_1 不含来自 α 的匹配事件对，而且到达与 w 相同的信道状态。我们通过对 w 中来自 α 的匹配事件对数量进行归纳来证明该断言。在基例中，设 $w_1 = w$ 和 $\alpha_1 = \alpha$。在归纳步骤中，设 $\alpha = \gamma_1 \cdot \mathrm{r} \to \mathrm{s} : m \cdot \gamma_2$，使得 $\mathrm{split}(\gamma_1 \cdot \mathrm{r} \to \mathrm{s} : m){\Downarrow_{\Sigma_r}} \leq w{\Downarrow_{\Sigma_r}}$，且 $\mathrm{split}(\gamma_1 \cdot \mathrm{r} \to \mathrm{s} : m){\Downarrow_{\Sigma_s}} \leq w{\Downarrow_{\Sigma_s}}$。设 w_r 是 w 关于 γ_1 对 r 的最大前缀，使得 $w = w_\mathrm{r} \cdot \mathrm{r} \triangleright \mathrm{s}!m \cdot u_\mathrm{r}$。设 w_s 是 w 关于 γ_1 对 s 的最大前缀，使得 $w = w_\mathrm{s} \cdot \mathrm{s} \triangleleft \mathrm{r}?m \cdot u_\mathrm{s}$。由 (F2) 可知，$w_\mathrm{r} \cdot \mathrm{r} \triangleright \mathrm{s}!m \leq w_\mathrm{s}$，并且我们可以写作 $w = w_\mathrm{r} \cdot \mathrm{r} \triangleright \mathrm{s}!m \cdot u \cdot \mathrm{s} \triangleleft \mathrm{r}?m \cdot v$，其中 $\mathrm{s} \triangleleft \mathrm{r}?m$ 不出现在 u 中。因此，$w_\mathrm{s}uv$ 与 $\gamma_1\gamma_2$ 合规，包含的来自 α 的匹配事件少一个，并且根据 (B8)，到达与 w 相同的信道状态。由此，根据归纳假设即可得出该断言。

由于 $w \cdot \mathrm{q} \triangleleft \mathrm{p}?m$ 是信道合规的，且 w_1 与 w达到相同的信道状态，因此 $w_1 \cdot \mathrm{q} \triangleleft \mathrm{p}?m$ 是信道合规的。

在第二步中，我们从 w_1 和 α_1中移除未匹配的发送事件。

断言 2. 存在 w_2 和 α_2，使得 w_2 与 $\alpha_2\beta$合规，w_2 不包含来自 α 的匹配事件对，也不包含来自 α的未匹配发送，并且 $w_2 \cdot \mathrm{q} \triangleleft \mathrm{p}?m$ 符合信道合规性。我们根据 w_1 中来自 α_1的未匹配发送数量，对该断言进行归纳证明。在基例中，设 $w_2 = w_1$ 和 $\alpha_2 = \alpha_1$。在归纳步骤中，设 $\alpha_1 = \gamma_3 \cdot \mathrm{r} \to \mathrm{s} : m \cdot \gamma_4$，使得 $\mathrm{split}(\gamma_3 \cdot \mathrm{r} \to \mathrm{s} : m){\Downarrow_{\Sigma_r}} \leq w{\Downarrow_{\Sigma_r}}$，并且相对于 γ_3，w中关于 r 的最大前缀，是相对于 α_1中所有未匹配发送事件而言最长的前缀。换言之，我们找到 w_1中来自 α_1 的未匹配发送事件最靠右的一次出现。首先，我们证明 $\mathrm{r} \to \mathrm{s} : m \neq \mathrm{p} \to \mathrm{q} : m$。由于 $w{\Downarrow_{\Sigma_q}} = \mathrm{split}(\alpha){\Downarrow_{\Sigma_q}}$，且 w 满足 (F2)，因此 $\mathrm{split}(\alpha){\Downarrow_{\Sigma_p}} \leq w{\Downarrow_{\Sigma_p}}$。于是，所有

来自 p 到 q 规定的发送事件 α 均未匹配。我们可以拆分 $w_1 = u_1 \cdot \mathtt{r} \triangleright \mathtt{s}!\ m \cdot u_2$，使得其中不存在来自 α_1 在 u_2中的未匹配发送。由 (B5) 可知，$u_1 \cdot u_2 \cdot \mathtt{q} \triangleleft \mathtt{p}?m$ 符合信道合规性，来自 α的未匹配发送少一个，并且与 $\gamma_3\gamma_4$合规。因此，该断言由归纳假设得证。

最后，令 α_3 为 α_2 移除所有不出现在 w_2 中的事件后所得。从上述两个步骤可知，显然 $\alpha_3 = \varepsilon$，因此 w_2 与 β信道合规，$w_2 \cdot \mathtt{q} \triangleleft \mathtt{p}?m$ 是信道合规的，且不包含 q 的任何事件。由此证明了 w_2 是 w'的见证。□

A.2 §7.1 的证明

引理 7.5。对于 $\mathbb{A} \in$ {点对点,同步广播, 包}，对任意满足 w 为 $\mathbb{A}$-信道合规且与 ρ一致的 $\rho \in \Gamma^*_{sync}$, $w \in \Sigma^*_{async}$，存在 $u \in \Sigma^*_{async}$，使得 wu 是信道合规的，并且 $wu \equiv_{\mathcal{P}}$拆分(ρ)。

证明。我们对 w 的长度使用归纳法证明该命题。在基例中，令 $u = \mathrm{split}(\rho)$。根据 (F3)，u 是信道合规的；根据构造，$u \equiv_{\mathcal{P}} \mathrm{split}(\rho)$。在归纳步骤中，令$w = w'x$。根据归纳假设，给定 u'，使得 $w'u'$是信道合规的，并且$w'u' \equiv_{\mathcal{P}} \mathrm{split}(\rho)$。令 p 为 x 中的活跃参与者，并令 u_1 为相对于 ε、针对 p 的 u' 的最大前缀。根据定义，$u_1{\Downarrow_{\Sigma_\mathtt{p}}} = \varepsilon$，且 u' 中紧接在 u_1 之后的下一个符号，其活跃参与者为 p。此外，由于 $w'x{\Downarrow_{\Sigma_\mathtt{p}}}$ 和 $w'u'{\Downarrow_{\Sigma_\mathtt{p}}}$ 都是 $\rho{\Downarrow_{\Sigma_\mathtt{p}}}$ 的前缀，u' 中紧接在 u_1 之后的下一个符号等于 x。因此，我们可以写作 $u' = u_1 y u_2$。令 $u = u_1u_2$。显然，u与 split(ρ) 按角色相等。为证明 $w'xu_1u_2$ 是信道合规的，我们需要以下关于所考虑的各种网络架构的事实。令 $v, u_1,$ $u_2 \in \Sigma^*_{async}$和 $z \in \Sigma_{async}$。

断言 *1.* 如果 vu_1zu_2 是点对点、同步广播或 包信道合规，$z = \mathtt{p} \triangleright \mathtt{q}!\ m$ 且$u_1{\Downarrow_{\Sigma_\mathtt{p}}} = \varepsilon$，则 vzu_1u_2分别是 点对点、同步广播或包信道合规。

断言 *2.* 如果 vu_1zu_2 是点对点或包信道合规，$z = \mathtt{q} \triangleleft \mathtt{p}?\ m$ 且$u_1{\Downarrow_{\Sigma_\mathtt{q}}} = \varepsilon$，则 vzu_1u_2 分别是 点对点和包信道合规。

当 z 是接收事件时，要满足同步广播通道合规性，我们对 u_1 要求稍强一些的假设。

断言 *3.* 如果 vu_1zu_2 是同步广播通道合规，$z = \mathtt{q} \triangleleft \mathtt{p}?\ m$，$u_1{\Downarrow_{\Sigma_\mathtt{p}}} = \varepsilon$，$u_1{\Downarrow_{\mathtt{p}\triangleright\text{-}!\text{-}}} = \varepsilon$，并且 $v{\Downarrow_{\text{-}\triangleleft\mathtt{p}?\text{-}}} \cdot z \preceq v{\Downarrow_{\mathtt{p}\triangleright\text{-}!\text{-}}}$，则 vzu_1u_2 是同步广播通道合规。

当 $z = \mathtt{q} \triangleleft \mathtt{p}?m$ 时，我们根据以下事实进行推理：$w' \cdot \mathtt{q} \triangleleft \mathtt{p}?m$ 是同步广播信道合规的，$w'u_1 \cdot \mathtt{q} \triangleleft \mathtt{p}?m$ 是同步广播信道合规的，并且 q 在 u_1 中没有事件；由此可知，q 不会在 u_1 中接收来自 p 的消息，因此没有其他参与者会在 u_1 中接收来自 p 的同步广播消息。因此，重排 z 到 u_1 之前所需的假设均已满足。□

A.3 复杂度

定理 5.5 立即给出了一个判定程序，用于判定以下网络架构上的有限协议是否可实现：$\mathbb{A} \in$ {点对点、同步广播、多播、monob、包}。Li 等人 [54] 证明了，在点对点网络上，有限全局协议的可实现性问题是 点对点 co-NP 完全的。我们首先考察各网络的可用性谓词之间的关系，以说明这一结果可推广至所考虑的所有齐次网络架构。根据我们关于如何判定 §7 中广义一致性条件的讨论，只需论证：(a) 可用性 对于 同步广播和 包 是 co-NP 完全的；(b) 可用性′ 对于多播 和 monob 是 co-NP 完全的。显然，上述问题的见证都可以在多项式时间内验证。

Li 等人 [54] 给出的 co-NP 下界证明通过将 3-SAT 归约到可实现性来实现。该证明假设一个 3-SAT 实例 $\varphi = C_1 \wedge \ldots \wedge C_k$，其中包含变量 $x_1, \ldots, x_n$以及文字 L_{ij}，表示子句 C_i中的第 j个文字，且 $1 \le i \le k$和 $1 \le j \le 3$。基于此，

构造一个全局协议$\mathcal{S}\varphi$，使得 ϕ 不可满足，当且仅当$\mathcal{S}\varphi$ 可实现。我们在图7中用图示概述这一构造。我们证明，对其稍作修改后，同一套下界构造也适用于其余网络架构。

该构造依赖两个构件：S_X，一种对变量 $x_1, \ldots, x_n$ 的变量赋值进行编码的构件（图 8）；以及 S_C，一种对子句 $C_1, \ldots, C_k$ 的文字选择进行编码的构件（图 9）。图 7 中高亮显示的消息对于 q_2 中的参与者 q 可用，当且仅当 φ 可满足。因此，协议 $\mathsf{S}\varphi$ 不可实现，当且仅当高亮显示的消息对于 q_2 中的参与者 q 可用；这又当且仅当 φ 可满足。为了说明该构造可以推广到包网络和发送方箱网络，只需分析每个构件中出现的唯一消息接收。因此，图 8 和图 9 分别展示了每个构件中足以确立我们新的复杂性结果的最小部分。关于该构造的完整细节，请参阅 [54]。

首先，我们考虑包网络架构。如§7所述，在点对点网络中可用的任何消息在包网络中也可用。因此，我们只需证明$\mathcal{S}\varphi$ 的其余部分可实现，这相当于检查是否还会出现其他包接收一致性违例。

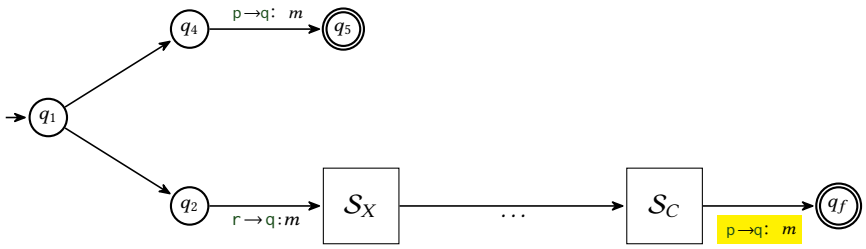


图 7. 从 [54] 得出的可实现性 3-SAT 归约示意图。

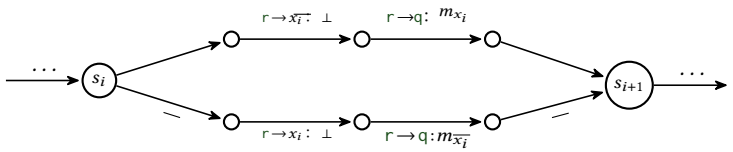
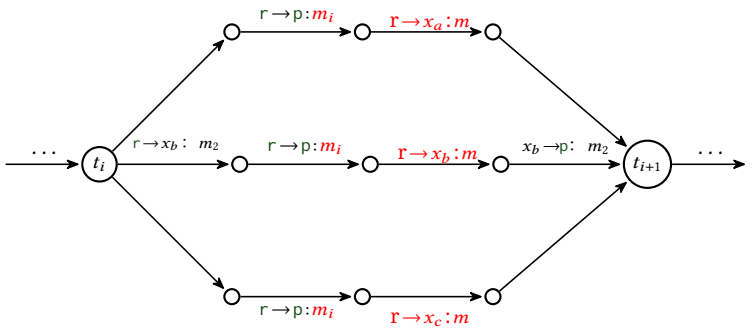


图 8. 修改后的变量赋值构造 S_X 的示意图。



Fig。9. 修改后的子句选择构件示意图S_C。修改之处以红色突出显示。

参与者 r 不接收消息，因此可以忽略。与点对点接收一致性不同，包接收一致性还会约束来自同一发送方的接收事件对。对于 S_X 中的变量参与者，每个参与者接收的要么是来自 r 的 $\perp$，要么是来自 q 的 $\top$，随后再接收一条 m消息。因此，包接收一致性得到满足。检查 S_C 可知，每个变量参与者只接收一种消息，即来自 r 的 m；若收到该消息，它就会向 p 发送一条 m 消息。因此，包接收一致性也得到满足。参与者 p 在 S_X 中不参与，但在 S_C 中会接收来自 r 的m 消息，通知它预期将收到某个变量参与者发来的消息。原始编码使用来自 r 的相同消息载荷来通知 p 预期将收到一条消息，但我们可以修改构造，让 r 向 p 发送一条消息，精确编码 p 应预期从哪个变量参与者接收消息。这样便消除了 p 原本可能出现的包接收一致性违规，因为 r 和变量参与者可能会超越 p 的接收事件。最后来看参与者 q：它在 S_C 中不参与，只在 S_X 中参与；q 恰好接收来自 r 的 n条消息，这些消息构成 n种二元选择：接收 m_{x_i}和 $m_{\bar{x_i}}$，其间穿插着 q 的发送事件。因此，我们可以得出结论：修改后的构造不可实现，当且仅当 S_φ 中满足 $\mathrm{bagavail}_{\mathsf{p},\mathsf{q},\{\mathsf{q}\}}(m, q_3)$，当且仅当 φ 可满足。

接下来，我们考虑同步广播网络架构。由于点对点可实现性蕴含同步广播可实现性，因此在这种情况下，我们只需分别证明可用性$_{\mathsf{p},\ \mathsf{q},\ \{\mathsf{q}\}}(m,\ q_3)$对同步广播可用性情形成立。如上所示，来自同一发送方、发往不同接收方且被阻塞的消息可能会破坏发送箱可接收性，因此我们需要检查子协议p 中是否存在此类发往其他接收方的消息：$\mathcal{S}_X$ 和$\mathcal{S}_C$。显然不存在此类消息，因此发送箱可实现性仍然成立。

对于monob 和多播，我们证明当且仅当可用性′ $_{\mathsf{p},\ \{\mathsf{r},\ \mathsf{p}\}}(q_2)$ 成立时，φ可满足。在构件$\mathcal{S}_X$中，同一运行过程中，同一接收方不会发生因果独立的接收。在构件$\mathcal{S}_C$中，增加额外的消息交换 $\mathsf{r} \to x_a{:}m$可确保来自r 和 x_a的消息无法在p的邮箱中独立地重新排序。因此，该构造确保广义一致性条件唯一可能的违反之处在于高亮状态 q_2中消息的可用性。

此外，我们认为，该构造可以调整为具有有向选择的多方会话类型，即有限全局协议的一个语法定义片段，其定义如下：

全局多方会话类型。 多方会话类型的全局类型 [42] 定义如下文法：

$$G ::= 0 \mid \sum_{i \in I} \mathsf{p} \to \mathsf{q}{:}m_i.G_i \mid \mu t.\ G \mid t$$

w其中 p、q 遍历 $\mathcal{P}$，m_i遍历有限集合$\mathcal{V}$，而 d遍历递归变量集合。

请注意，图 7 中的顶层选择满足有向选择，因此我们只需修改赋值构件和子句选择构件。我们在图 10 和图 11中描绘了为满足有向选择而修改后的构件。从有向选择全局协议到有向选择多方会话类型的编码遵循 [55], ，详情请参阅其附录。

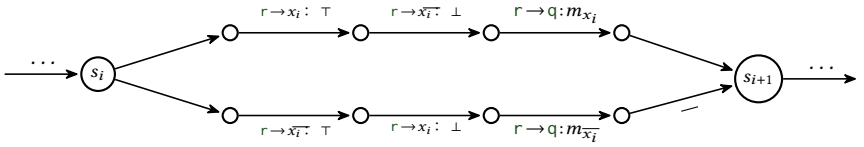


图10. 插图　图。修改部分以红色突出显示。

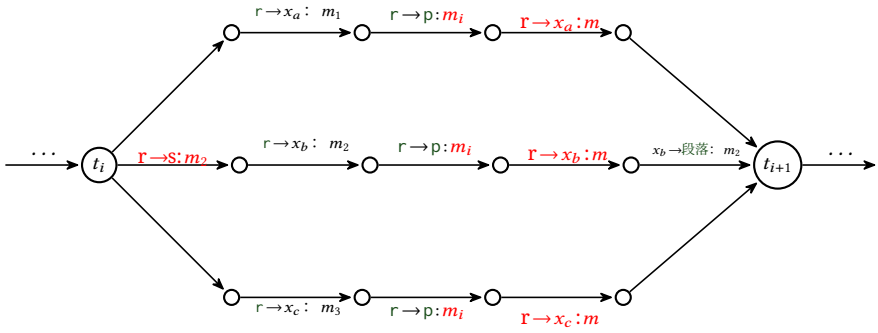


图 11. Illustration of 有向选择子句选择组件 S_C。修改部分以红色突出显示 .